Universidade de Aveiro
2017

Departamento de Eletrónica, Telecomunicações e Informática

Jailson Eduardo Ramos Brito Évora

Reengenharia dos Workflows do Sistema de Informação da Justiça de Cabo Verde

Universidade de Aveiro
2017

Departamento de Eletrónica, Telecomunicações e Informática

Jailson Eduardo Ramos Brito Évora

Reengenharia dos Workflows do Sistema de Informação da Justiça de Cabo Verde

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, realizada sob a orientação científica do Doutor Cláudio Teixeira, Equiparado a Investigador Auxiliar e do Doutor Joaquim de Sousa Pinto, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

Aos meus pais

o júri

presidente

Prof. Doutor Joaquim Arnaldo Carvalho Martins

Prof. Doutor André Frederico Guilhoto Monteiro
Professor Auxiliar Convidado do Instituto Superior Miguel Torga

Prof. Doutor Cláudio Jorge Vieira Teixeira
Equiparado a Investigador Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos

A Deus e a minha família. Igualmente um especial agradecimento ao Professor Joaquim Sousa Pinto e ao Professor Cláudio Teixeira e a todos que durante esse percurso ajudaram-me a incutir novos conhecimentos e a ultrapassar os mais diversos obstáculos.

palavras-chave

Cabo Verde, Justiça, Processo Penal, Modelação de Processos de Negócios, Windows Workflow Foundation, Automatização de processos, Arquitetura Orientada a Serviços (SOA)

resumo

Nesta dissertação, apresenta-se a reestruturação do sistema de workflows do Sistema de Informação da Justiça de Cabo Verde (SIJ). Esta restruturação irá permitir a integração de novas ferramentas para a modelação e definição dos processos de negócios, bem como, a integração do novo motor de workflows que é o centro dos sistemas de workflows.
Além da integração das ferramentas para o desenvolvimento e manutenção da lógica de negócio do SIJ, também este trabalho permitirá, a partir da reutilização de códigos anteriores, ter serviços granulares e independentes, com a retirada da lógica de dentro de workflows, para componentes que disponibilizam interfaces de serviços possíveis de serem acedidos independentemente da plataforma, localização ou ambiente que os rodeia.
No final, ter-se-á um “novo” sistema de workflows capaz de ser escalável, dinâmico e que otimize a lógica de negócio por detrás da tramitação e desmaterialização dos processos nos tribunais de Cabo Verde.

keywords

Cabo Verde, Justice, Criminal Proceedings, Business Process Modeling, Windows Workflow Foundation, Process Automation and Process Execution, Service Oriented Architecture

abstract

In this dissertation, the restructuring of the workflow system of the Cabo Verde Justice Information System (SIJ) is presented. This restructuring will allow the integration of new tools for modeling and definition of business processes as well as the integration of the new workflow engine which is the center of the workflow systems.

In addition to the integration of the new notation and tools for the development and maintenance of SIJ business logic, this work will also allow, along with the reuse of inherited coding to have granular and independent services, with the withdrawal of logic from within workflows, for components that provide service interfaces that are accessible regardless of the platform, location, or environment that surrounds them.

In the end, there will be a "new" workflow system capable of being scalable, dynamic and that optimizes the business logic behind the processing and dematerialization of the processes in the courts of Cabo Verde.

Índice

Índice Figuras

Índice de Tabelas

Índice de Código

Acrónimos

API	Interface de programão de aplicações
BPEL	Business Process Execution Language
BPMN	Business Process Model and Notation
BPModeling	Modelação de processos de negócios
BPMS	Sistema de gestão de processos de negócios
CORBA	Common Object Request Broker Architecture
CPU	Unidade de processamento central
DGCI	Direção Geral de Contribuição e Impostos de Cabo Verde
DGTR	Direção Geral de Transportes Rodoviários
DJE	Diário Judicial Eletrónico
DMBS	Sistemas de gestão de base de dados
EAI	Integração de aplicações empresarias
EPC	Cadeia de processos dirigido pelos eventos
GUID	Identificador único global
ITIL	Information Technology Infrastructure Library
JSON	JavaScript Object Notation
MOM	Message-Oriented-Middleware
MP	Ministério Público
OACV	Ordem dos Advogados de Cabo Verde
OMG	Object Management Group
ORDBMS	Sistema de gestão de base de dados de objetos relacional
PN	Processos de negócios
RNI	Registo Nacional de Identificação
SI	Sistemas de Informação
SIIC	Sistema de Informação de Investigação e Criminalidade
SIJ	Sistemas de Informação da Justiça de Cabo Verde
SIPC	Sistemas de Informação do Processo Cível
SIPP	Sistemas de Informação do Processo Penal

SO	Orientação a serviços
SOA	Arquitetura orientada a serviços
SOAP	Simple Object Access Protocol
TI	Tecnologias de Informação
UDDI	Universal Description, Discovery, and Integration
UML	Unified Modeling Language
WCF	Windows Communication Foundation
WF	Windows Workflow Foundation
WfMC	Workflow Management Coalition
WfMS	Sistemas de gestão de *workflows*
WS	Web Service
WSDL	Web Service Definition Language
XML	EXtensible Markup Language

1. Introdução

Face à necessidade de agilizar e melhorar a eficiência no setor da justiça cabo-verdiana, em 2008, o Ministério da Justiça de Cabo Verde (MJ), juntamente com o Conselho Superior da Magistratura Judicial (CSMJ) e o Conselho Superior do Ministério Público (CSMP) iniciaram o planeamento para o desenvolvimento de um sistema de informação, designado de Sistema de Informação da Justiça de Cabo Verde (SIJ).

Por natureza, o SIJ é um sistema onde a lógica de negócio está sujeita a constantes mudanças, face a alterações constitucionais e no funcionamento judiciário. Essa lógica é constituída por uma série de atividades que devem ser completadas para realizar uma unidade de trabalho, fornecendo representações de modelos existentes, que podem ser conduzidos usando as tecnologias de *workflow*.

As tecnologias de *workflow* permitem a representação de um modelo de programação declarativo ao invés dos tradicionais procedimentais, promovendo uma clara separação entre o que fazer e quando fazê-lo, permitindo alterar o "quando" sem afetar "o quê" [1]. Este igualmente permite aos intervenientes terem a visão geral dos processos modelados de forma bidimensional, facilitando a transparência na modelação.

Assim sendo, advindo da utilização das tecnologias de *workflows* e os benefícios inerentes, é possível desenvolver sistemas que reduzam a lacuna existente entre a parte técnica da programação e os requisitos que são criados, proporcionando um nível de abstração e automatização dentro dos processos de negócios [2]. Estes processos de negócios, tecnologicamente têm sido suportados pela utilização de tecnologias de *workflow*, adicionado ao surgimento de um novo paradigma, orientado a serviços (SO) e à arquitetura inerente (SOA).

1.1 Motivação

O SIJ em si só é um sistema reativo por natureza, ou seja, aguarda por um período indefinido que um evento aconteça. Esses eventos despoletam o processamento por parte do sistema, onde o desempenho e a escalabilidade são afetados, pois enquanto aguarda por eventos externos, está igualmente a consumir tempo de CPU e memória RAM. Desta forma, para fazer face a esta característica, a equipa de desenvolvimento na fase inicial da conceção, fez o uso do Windows Workflow Foundation (WF) para modelar e gerir a lógica de negócio do SIJ. O WF é uma estrutura extensível que fornece uma interface de programação (API) e ferramentas para o desenvolvimento e execução de aplicações baseados em *workflows* na plataforma Microsoft [3][4].

Inicialmente o sistema de *workflows* foi implementado na versão 3.0 e 3.5 do WF, onde é permitido modelar utilizando *workflows* sequenciais e máquinas de estados. Face às necessidades e características do SIJ, utilizaram-se as máquinas de estado para a modelação de todos os *workflows* referentes ao processo penal e ao processo cível. Máquinas de estado são um tipo de *workflow* que depende de eventos externos para a transição entre estados, normalmente associados a decisões humanas.

Tal como toda a tecnologia, sendo WF não uma exceção, este está sujeito a atualizações e reformulações. Nos meados de 2009, a Microsoft disponibilizou uma nova versão, designada de WF 4.0, restruturando por completo a arquitetura (modelo de programação, tempo de execução e novas ferramentas) [5]. Além da versão WF 4.0 não ser compatível com a anterior 3.5, este também não contém as máquinas de estado, alegando que fluxogramas, um novo recurso (novo tipo de *workflow*) adicionado a esta versão, servia para o substituir. Sendo assim, não foi possível fazer o upgrade da versão 3.5 para a versão 4.0 no SIJ.

Face a supressão das máquinas de estado na versão 4.0, a comunidade de programadores ligados ao WF apelaram à reintegração desse recurso, vindo a Microsoft disponibilizar a nova versão, juntamente com o novo .Net Framework 4.5. A nova versão WF 4.5, voltou a incorporá-lo, mas não na sua forma inicial. Por conseguinte o sistema de *workflows* do SIJ encontra novamente um outro contratempo, pois esta nova versão não suporta uma série de funcionalidades, como (re)criar um determinado processo em qualquer ponto do *workflow*, a possibilidade de administrativamente alterar o ponto de execução atual do processo para um ponto anterior ou posterior no *workflow* e também a não possibilidade de ter estados dentro de estados.

1.2 Problema

Para além dos contratempos referidos na secção 1.1, acabou-se igualmente por identificar alguns pontos menos positivos da utilização do WF, fruto da experiência acumulada pela equipa de desenvolvimento ao longo destes anos de utilização. Constatou-se que o WF está propenso a erros, onde cada *workflow* quando criado tem associado a si uma instância única. Esta instância, em caso de acontecer um erro de execução, não permite recriar com a mesma instância, ou seja, *workflows* corrompidos não podem ser recuperados em algumas versões. Em termos de desempenho, este requer recursos computacionais consideráveis, tanto na modelação, na fase de desenvolvimento, bem como, depois de ser colocado em produção. Outro aspeto tem a ver com a curva de aprendizagem, acabando por não ser das mais fáceis, requerendo um esforço adicional tanto na manutenção como na implantação do sistema. Igualmente na maioria das versões, não é possível a compatibilidade entre estes, na medida em que se for preciso atualizar *workflows* criados e persistidos, a instância antiga é incompatível, apesar de existirem mecanismos para efetuar a migração, nem todos adaptam e esses mesmos mecanismos não são eficientes o suficiente.

Decorrente destas desvantagens em utilizar a tecnologia WF, o SIJ tem-se deparado com surgimento de processos com fases duplicadas, *workflows* persistidos com fases diferentes da do processo, utilizadores únicos com acessos ativos duplicados ao processo, *workflows* a abortar execuções e consumo excessivo de recursos computacionais pelos serviços de *workflows*. Parte destes problemas advêm da execução de código no *workflow*, que tem duplicado as execuções, em circunstâncias não completamente conhecidas.

Tal como referenciado na secção 1.1, com a evolução do WF, foi perdida a capacidade de (re)criar um determinado processo em qualquer ponto do *workflow*, bem como a possibilidade de administrativamente, alterar o ponto de execução atual do processo para um ponto anterior ou posterior. Também, dada a complexidade dos processos judicias, há a necessidade de eventos distintos poderem ser despoletados em mais do que um estado. Este comportamento é possível na versão 3.5 com a funcionalidade de estados dentro de estados, mas com as alterações feitas nas versões posteriores, este tornou-se impraticável.

Desta forma, algumas destas necessidades ((re)criar um determinado processo em qualquer ponto e alterar o ponto de execução atual) não se coadunam com os motores tradicionais, ainda que existam na WF 3.5 em utilização. Dada a necessidade de atualização do motor de *workflows* (para resolver algumas das questões identificadas) e dada a supressão das funcionalidades aqui referidas, tornou-se urgente a procura de novas soluções. O trabalho de pesquisa dessas soluções revelou que os motores de processamento de *workflows* não preveem a criação de instâncias em pontos arbitrários nem a execução de transições não previstas no desenho do *workflow*. Assim, tornou-se necessária a criação de um motor de *workflows* que satisfizesse estas necessidades. Esse trabalho foi desenvolvido por um dos elementos da equipa, em articulação com o trabalho referente a esta dissertação de mestrado.

Com o novo motor de *workflows*, torna-se necessário efetuar mudanças no paradigma utilizado no sistema de *workflows* do SIJ, onde este trabalho propõe a reutilização e adaptação de alguns componentes da abordagem anterior numa nova arquitetura. A nova alternativa deve ser capaz de minimizar as desvantagens da arquitetura anterior, tendo em consideração que a lógica de negócio num sistema judicial está sujeita a processos de longa duração, devendo lidar com estados e qualquer coexistência de mensagens ao receber um evento.

1.3 Objetivos

Decorrente dos problemas abordados até então, a equipa de desenvolvimento sentiu-se a necessidade de criar e implementar uma nova abordagem para os sistemas de *workflows*. Este "novo" sistema deve permitir a equipa desenhar, automatizar e otimizar processos de negócios baseados em vários serviços fracamente acoplados, onde este significa uma unidade de um programa que atende a um processo de negócios [6] [7]. Esta abordagem deve conciliar o motor de *workflows,* o conversor de definição de processos, bem como a notação definida pela equipa para a modelação de processos.

Esses objetivos devem ser alcançados primeiramente com a reutilização do código das atividades de WF 3.5, para serviços granulares e independentes, retirando a lógica de negócios de dentro dos *workflows*, ficando este apenas com a responsabilidade de conduzir os processos de negócios.

Igualmente com esses serviços granulares definidos e implementados, deve-se efetuar a integração desse novo componente na arquitetura do sistema. Esta integração irá exigir a criação de componentes auxiliares e efetuar alterações em projetos existentes. Essas alterações devem ser cautelosas, visto que esta nova abordagem irá basear-se somente no processo penal deixando os outros tipos de processos judicias para trabalhos futuros. Desta forma, os tipos de processos (cível, recurso, *habeas corpus*) que não vão ser comtemplados devem permanecer utilizando a tecnologia WF.

1.4 Sistema de Informação da Justiça de Cabo Verde

Tal como apresentado na secção 1, o Sistema de Informação da Justiça de Cabo Verde é um projeto financiado pelo MJ tendo como parceiros o CSMJ, CSMP e a Universidade de Aveiro, com o objetivo de informatizar e desmaterializar os processos judiciais nas secretarias dos tribunais de Cabo Verde.

O SIJ consiste numa aplicação web onde cada utilizador possui credencial de acesso único mediante o papel que desempenha no sistema judicial cabo-verdiano, desde Procurador-Geral da Republica, Juízes, Inspetores/Procuradores e Oficiais de Justiça. Esses utilizadores podem trabalhar com processos do âmbito Cível e/ou Penal mediante as permissões e o perfil destes.

Em termos de componentes o SIJ é constituído pelo Sistema de Informação do Processo Penal (SIPP), Sistema de Informação do Processo Cível (SIPC), Diário Judicial Eletrónico (DJE), Sistema da Ordem dos Advogados de Cabo Verde (OACV), *IT Operational Portal ITIL* (gestor de incidentes, pedidos de utilizadores), interfaces com serviços externos e Identificador Único.

Para além dos componentes anteriores, este possui conexões com outros serviços da rede do Estado cabo-verdiano, entre eles, o Sistema de Informação de Investigação e Criminalidade (SIIC), a Direção Geral de Contribuição e Impostos (DGCI), o Registo Nacional de Identificação (RNI), a Direção Geral de Transportes Rodoviários (DGTR) e o Registo Predial (Figura 1).

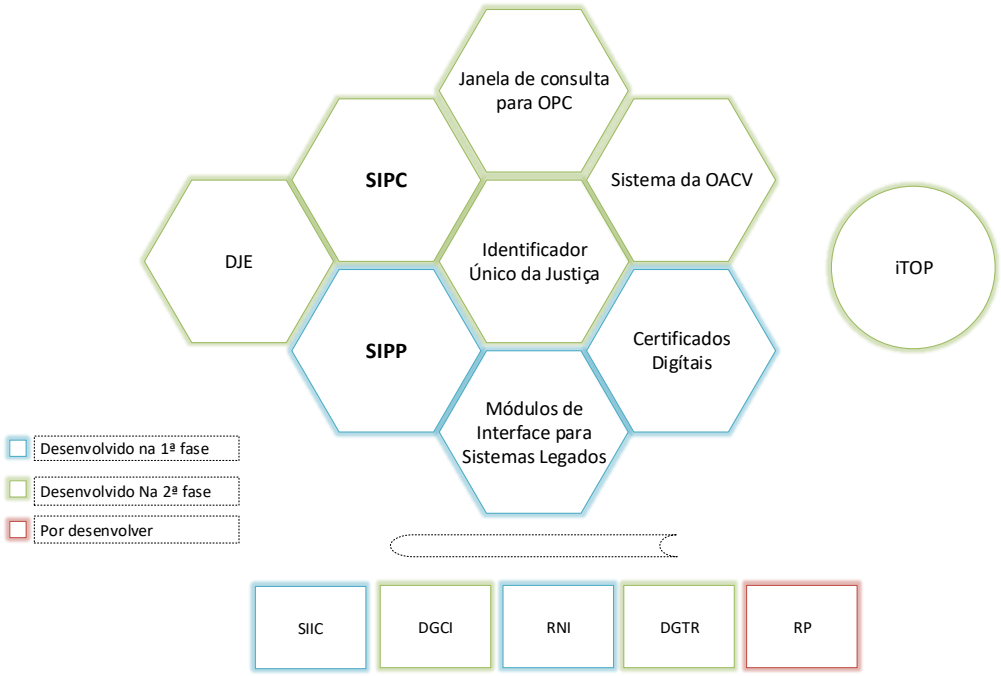


Figura 1: Arquitetura conceptual dos componentes do SIJ [8].

Tecnicamente falando, o SIJ é um sistema integrado e único onde tanto o SIPP e o SIPC partilham a mesma estrutura lógica, estando na mesma solução. O SIPP e SIPC são aplicações que assentam na arquitetura orientado a serviços e numa arquitetura de multicamadas onde os modelos de dados e a vista estão separados em projetos diferentes. Por outro lado, toda lógica de negócio é tratada pelo sistema de *workflows*, decorrente

das definições do sistema judicial onde as diferentes fases que compõem um processo, já encontram previamente identificados e modelados.

O SIJ foi concebido utilizando a base tecnológica .NET Framework, sendo esta uma tecnologia que suporta a construção e execução de aplicações e Web-Services (WS) [9]. .NET continua a desempenhar um papel central no desenvolvimento de novas funcionalidades e tarefas de manutenção do sistema. Sendo Windows Communication Foundation (WCF) um dos componentes da plataforma .NET, a equipa de desenvolvimento faz o uso desta tecnologia para a conceção dos WS, assentes no paradigma orientados a serviços, tendo o WS como uma das características, a possibilidade de envio de mensagens assíncronas de um ponto final (fornece aos clientes do serviço acesso à funcionalidade oferecida [10]) do serviço para outro [11] [12] [13].

1.5 Contribuição

Nesta dissertação, apresenta-se a reestruturação do sistema de workflows do Sistema de Informação da Justiça de Cabo Verde (SIJ). Esta restruturação irá permitir a integração de novas ferramentas para a modelação e definição dos processos de negócios, bem como, a integração do novo motor de workflows que é o centro dos sistemas de workflows.

Além da integração das ferramentas para o desenvolvimento e manutenção da lógica de negócio do SIJ, também este trabalho permitirá, a partir da reutilização de códigos anteriores, ter serviços granulares e independentes, com a retirada da lógica de dentro de workflows, para componentes que disponibilizam interfaces de serviços possíveis de serem acedidos independentemente da plataforma, localização ou ambiente que os rodeia.

No final, ter-se-á um "novo" sistema de workflows capaz de ser escalável, dinâmico e que otimize a lógica de negócio por detrás da tramitação e desmaterialização dos processos nos tribunais de Cabo Verde.

1.6 Organização da dissertação

A presente dissertação está dividida em seis capítulos, sendo este o capítulo 1 que faz a introdução ao tema em estudo, apresenta a motivação e problemas. Igualmente apresenta os objetivos, a contribuição, faz o enquadramento do SIJ, apresentando os diversos componentes que o constitui.

No segundo capítulo são apresentados os conceitos teóricos, procurando construir uma referência em torno da automatização de processos, modelação de processos, serviços usando SOA integrados a processos, mecanismos de segurança para WS bem como sistemas de *workflows* e as tecnologias inerentes a este.

O terceiro capítulo aborda os principais problemas da arquitetura ainda em uso, apresenta a tipologia dos *workflows* do SIJ, os principais componentes do sistema de *workflows* e descreve a interação entre os vários serviços utilitários.

O quarto capítulo apresenta o desenho dos serviços granulares, a arquitetura utilizada bem como, as configurações efetuadas para o seu funcionamento. Estes serviços foram desenvolvidos com o objetivo de

retirar a lógica de dentro dos *workflows* utilizados anteriormente para pequenos componentes que podem ser referenciados independentemente da localização.

O quinto capítulo apresenta o caso de estudo com a adaptação à nova arquitetura de *workflows*, conciliando a API do motor de *workflows* ao conversor de definição de processos. Também aborda a notação estabelecida para a modelação dos processos e as alterações realizadas nos componentes já existentes a fim de integrar os serviços granulares a arquitetura do sistema de *workflows*, ficando este com responsabilidade de fazer o "*drive*" dos processos.

O sexto capítulo apresenta as conclusões da presente dissertação e uma proposta de trabalho futuro que enquadre os processos judiciais cíveis e a implementação de mecanismos que garantam a integridade e direitos de acesso as interfaces dos serviços granulares.

2. Contextualização técnica

Neste capítulo abordam-se conceitos técnicos que servirão como suporte ao presente estudo. Assente nestes conceitos será possível construir modelos de processos que capturem a relação lógica e temporal entre as atividades, objetos e os recursos necessários para a disciplina de modelação.

Tendo conhecido os conceitos referentes a modelação, torna-se pertinente apresentar a automatização de processos, começando primeiramente com a definição do conceito chave, *workflow* e fazer uma retrospetiva deste conceito relacionado ao mundo das TI. Ainda na automatização de processos são apresentados dois tipos de sistemas de *workflows* comummente utilizados e as áreas funcionais desses sistemas.

Para finalizar o capítulo, aborda-se a arquitetura orientada a serviços, os mecanismos de segurança e o modelo de referência proposto pela Workflow Management Coalition (WfMC), que é a base para a construção de sistemas de *workflows*. Nessa abordagem ao modelo de referência, são apresentados os componentes e a arquitetura de Sistemas de Gestão de Workflows (Workflow Management Systems - WfMS). Igualmente é feita a comparação da arquitetura de WfMS proposto por este padrão com a nova tecnologia no mercado que teve como base os primeiros sistemas de *workflows*.

2.1 Processos

Nas organizações e TI, o processo muitas vezes pode ser entendido como sendo um conjunto limitado de atividades que são realizadas em resposta a algum evento, a fim de gerar uma saída [14], podendo ser muito simples ou extremamente complexa [15]. É um conceito um tanto mais ambíguo com significados diferentes, dependendo do contexto em que é usado [16].

Desta forma, quando se contextualiza no seio das organizações e TI, é importante compreender, que ter um evento de entrada a ser transformado em uma saída é designado muitas vezes como sendo um modelo de transformação simples. Este modelo de transformação simples, constitui a base para a assim chamada visão de processo de uma organização. Numa primeira perspetiva, este modelo simplista oculta os detalhes da transformação, sendo desta forma necessário para se conseguir uma análise e um desenho detalhado, precisando ir mais além, entrando na chamada "caixa negra", para ser possível conhecer os tipos de processos, as hierarquias e os determinantes da arquitetura em destaque (Figura 2) [16].

Figura 2: Modelo de transformação de um processo [15], [16].

O modelo de transformação recebe uma entrada e na "caixa negra*"* são efetuadas as transformações por um número indeterminado de tarefas ou atividades, acabando por despoletar uma saída das transformações realizadas pelo processo, tal como referido anteriormente na definição do processo.

Mas só isso não é suficiente para entender de todo o significado e a constituição de um processo, para tal, explorando mais detalhadamente a constituição da referida "caixa negra", acaba-se por deparar que este pode ser constituído por diferentes tipos de processos, onde muitas vezes são divididos em três tipos diferentes: acabando os **individuais** por serem realizados por indivíduos separados; os processos **verticais** ou funcionais, que estão contidos dentro de uma determinada unidade funcional; e por fim processos funcionais **horizontais**, que atravessam várias unidades [16], [17].

Igualmente é importante realçar que as estruturas do processo podem ser caracterizadas em cinco componentes ou elementos principais, sendo as entradas e saídas, as unidades de fluxo, a rede de atividades e os *buffers*, os recursos e a estrutura de informação.

2.2 Processos de negócios

O termo negócio pode ser entendido como uma entidade organizacional que insere recursos para fornecer aos clientes os produtos ou serviços desejados, englobando as empresas e os demais. Desta forma, um processo de negócios (P**N**) descreve como algo é feito em uma organização [16]. Este pode ser pensado como um conjunto de atividades de curta ou longa duração, invocadas numa sequência específica para atingir um objetivo de negócio específico [18], [19], [20], envolvendo etapas executadas por máquinas e por pessoas [21], [22].

Explorando o conceito aplicado a TI, normalmente uma linguagem de metalinguagem acaba por ser utilizada para descrever um processo de negócios, podendo invocar outros serviços de suporte ou conter outros componentes de serviços, como máquinas de estados, tarefas humanas, regras de negócios ou mapas de dados [22].

Sendo assim, da mesma forma que processos de negócios tem os seus princípios e conceitos, é importante fazer uma separação entre este e *workflows*. É comum deparar com confusões em torno desses dois conceitos. PN não deve ser confundido com *workflow*. Workflow é a forma como o trabalho é feito, sendo dependente do suporte dos processos de negócios, contudo direcionado pela lógica do processo e regras de orquestração [22].

2.3 Modelação de processos

Na modelação de processos existe uma disciplina específica, conhecida por Modelação de Processos de Negócios (*BPModeling*). Esta é entendida como sendo a disciplina que define e delineia práticas de negócios, processos [23], fluxos de informações e armazenamento de dados [22], [24]. Envolve o uso de uma notação como a Unified Modeling Language (UML) ou Business Process Model and Notation (BPMN) para capturar representações gráficas dos principais processos, *workflows*, entre outros aspetos relacionados com a lógica do negócio. Tem como principal objetivo desenvolver o modelo de processo, que irá definir o fluxo de processo existente em detalhe, sendo necessário responder às seguintes questões na sua implementação [7]:

- Qual é o resultado do processo de negócio?
- Que atividades são realizadas dentro do processo de negócios?
- Qual é a ordem das atividades?
- Quem realiza as atividades?
- Que documentos são trocados dentro do processo?
- Quais as fraquezas do processo, e como pode ser estendido no futuro?

Tendo em mente essas questões delineadas na especificação, é crucial garantir os traços a listar seguidamente, para obtenção de um bom modelo. É recomendável que o modelo seja [25]:

- Saliente - deve representar seletivamente aspetos relevantes para a tarefa em questão;
- Preciso - deve sistematizar com precisão o estado atual dos assuntos e evitar visões erróneas ou tendenciosas;
- Completo ainda que cauteloso - tão simples quanto possível;
- Compreensível - uma vez que se percebe o modelo, deve-se ser capaz de entendê-lo.

Tal como falar das questões e traços acima retratados, que devem ser seguidos na implementação desta disciplina, é imperativo também apresentar as técnicas mais utilizadas na modelação de processos. Desta forma, existem muitas técnicas que foram e vêm sendo testadas ao longo dos anos, sendo por vezes consideradas um pouco confusas, decorrente das muitas definições e várias maneiras de chegar ao fim proposto.

É importante conhecer as várias técnicas disponíveis para que a escolha se enquadre e adeque às características dos processos a modelar, facilitando a tarefa de modelação e que vá de encontro as necessidades. Desta forma apresenta-se essas técnicas (Tabela 1), sendo construída mediante pesquisa bibliográfica a diferentes fontes [18], [19], [24], [26] .

Tabela 1: Técnicas e notações para modelação de processos de negócios.

Técnicas	**Descrição**
BPMN	A intenção da BPMN na modelação de processos de negócios é muito semelhante à de UML para o desenho e análise orientados a objetos. Este visa identificar as melhores práticas das abordagens existentes e combiná-las em uma nova linguagem, geralmente aceite [18], [19], [23]. As linguagens antecessores a BPMN incluíram não apenas linguagens baseadas em gráficos e Petri-net, mas também diagramas de atividade UML e Event Driven Process Chain (EPC) [21]. Enquanto estas linguagens de modelação se concentram em diferentes níveis de abstração, variando de um nível de negócio a um nível mais técnico, BPMN visa suportar a gama completa de níveis de abstração, incluindo níveis de negócios e níveis de tecnologia de *software* [21], [27], [28].

	Este é estabelecido no documento de normas, que afirma: "*O principal objetivo do BPMN é fornecer uma notação que seja facilmente compreensível por todos os utilizadores de negócios, desde os analistas de negócios que criam os rascunhos iniciais dos processos até os programadores responsáveis pela implementação da tecnologia que irá executar esses processos e finalmente, para pessoas de negócios que irão gerir e monitorizar esses processos*" [20], [21], [29], [27], [28], [30].
Colored Petri-nets (CPN)	Linguagem gráfica que fornece primitivas básicas para construção de modelos a sistemas concorrentes [31]. Possui a característica de modelar eventos discretos bem como de comunicação e sincronização, combinando as capacidades das redes de Petri com as de uma linguagem de programação de alto nível [18], [19], [26], [32].
Data flow diagrams (DFD ou Yourdon´s *techinique*)	Desenvolvido no inico dos anos 60 por Larry Constantine e Ed Yourdon, os DFD podem ser visto como métodos de organização de dados a partir de seu estado bruto [26]. Tem como objetivo mostrar como circula os fluxos de informação de um lugar para outro, bem como o relacionamento destes com os utilizadores e o exterior. Podem ser usados para registrar as análises de processos como parte da documentação de projeto [33].
Event Driven Process Chain (EPC)	Notação informal para a representação de conceitos e processos de negócios em detrimento de seus aspetos formais ou sua realização técnica [26]. Faz parte de uma abordagem holística de modelação, denominada framework *Architecture of Integrated Information Systems* (ARIS) [33] desenvolvido por August-Wilhelm Scheer [18], [19].
Flow Chart	Por ser constituído por poucos padrões de símbolos, é facilmente compreendido. Também a sua simplicidade, fez com que tornasse numa ferramenta robusto e eficaz, sendo um dos percursores de muitas outras linguagens gráficas e notações, estando a ser utilizado durante muitos anos na modelação de processos. Utiliza um fluxo sequencial de ações e não suporta uma divisão das atividades. A notação BPMN pode ser considerado como uma versão avançada desta técnica [26].
Gantt Chart	Relaciona uma lista de atividades com uma escala temporal, podendo ser usado para representação de processos, apesar do seu foco residir na capacidade de monitorizar a situação atual, o cronograma do projeto e na alocação de recursos acabando por negligenciar a parte de modelação [33], [26].

Integrated Definition for Function Modeling (IDEF)	São basicamente conjuntos de métodos para modelação integrada direcionada a área de negócios e empresas. São utilizadas de acordo com diferentes aplicações, sendo, IDEF0, IDEF1, IDEF1X, IDEF2, IDEF3, IDEF4 e IDEF5 as partes mais importantes desta família de métodos [23]. No entanto, para a modelação de processos de negócios, as versões mais úteis são IDEF0 e IDEF3 [33], [26].
Object Oriented Methods (OO)	Método para analise, desenho e implementação a partir da técnica orientada a objeto, existindo muitas técnicas diferentes baseadas em OO [34]. A UML é considerada a linguagem padrão de modelação OO [26].
Petri Nets	As redes de Petri são das técnicas mais conhecidas para especificar processos de negócios formal e abstrata, como tal, uma importante base para as linguagens de processo. "Formal" significa que a semântica das instâncias de processo resultantes dos modelos especificados nas redes de Petri está bem definida e não é ambígua e "abstratas", porque não levam em consideração o ambiente de execução de um processo de negócios, de modo que todos os aspetos além das perspetivas funcionais e de processo não são cobertos [18], [19], [26].
Role-Activity Diagrams (RAD)	Notação abstrata para descrever um comportamento desejado dentro de uma organização. São muitas vezes funções organizacionais, podendo incluir sistemas de *software*, clientes e fornecedores [35]. Fornecem uma perspetiva diferente do processo e são particularmente úteis no apoio à comunicação [26].
Role-Interaction Diagrams (RID)	Embora ligeiramente mais complexos do que fluxograma, RIDs são bastante intuitivos de entender, fácil de ler, mas tendem a ser confusos e difícil de construir [26].
UML Ativity Diagram	UML é uma linguagem de modelação usada principalmente para especificação, visualização, desenvolvimento e documentação de sistemas de software [20]. Tornou-se também numa poderosa técnica de modelação de processos de negócios. Com 14 tipos diferentes, este oferece uma maneira flexível e poderosa de visualizar quase todos os processos de negócios [32], [23].
Workflow Diagram	Diagramas de *workflow* é uma abordagem de melhorar as redes tradicionais de Petri com conceitos e notações que facilitam a representação de processos de negócios, introduzindo restrições estruturais que se mostram úteis para os processos de negócios [18], [19].

	É mais do que uma técnica para modelar um processo. É uma metodologia para analisar e melhorar um processo, incluindo sua modelação. Compreende quatro etapas: recolha de informações, modelação de processos de negócios, modelação de *workflows*, implementação, verificação e execução [26].

2.4 Automatização de processos

Outro conceito chave que se torna pertinente abordar neste estudo, refere-se à automatização de processos. A automatização de processos pode ser abordada sob diferentes perspetivas, podendo referir-se à intenção de automatizar qualquer parte concebível do trabalho processual dentro de um processo empresarial [36]. Este abarca desde operações simples que fazem parte de uma única atividade de processo até a coordenação automatizada de processos inteiros e complexos [37].

Esta intenção de automatizar qualquer parte concebível do trabalho processual, leva ao conceito Workflow. Workflow é entendido como sendo um processo que é automatizado, no todo ou em parte, por um sistema de *software*, que transmite informações de um participante para outro, de acordo com a dependência temporal e lógica definida no modelo de processo subjacente [37].

Desta forma, nesta secção são abordados conceitos chave associados a automatização de processos, desde a definição de *workflow* à apresentação de conceitos e características deste.

2.4.1 Workflow e uma retrospetiva histórica

Antes de entrar em detalhe a respeito do conceito, importa fazer um levantamento temporal para enquadrar conceito com a realidade atual, bem como compreender as origens e surgimento desta "nova" abordagem no mundo das organizações e de TI.

Sendo assim, na década de 1990, surgiu a WfMC, fundada para definir padrões relacionados a *workflows*, por empresas do ramo de TI e dos negócios, bem como por utilizadores e académicos. Esta organização sentiu a necessidade de definir e padronizar o termo WfMS, entendendo-o como sendo um sistema que define, cria e gere a execução de *workflows* através do uso de *software*, capazes de interpretar o processo, interagir com participantes bem como quando necessário invocar o uso de ferramentas e aplicações de TI [18], [19].

Fazendo uma retrospetiva, constata-se que ao longo dos anos foram vários os marcos em torno do desenvolvimento de sistemas de informação. A partir de um resumo histórico, destacam-se [38]:

- **1965-1975: a decomposição das aplicações**, que compreenderam aplicações separadas por camadas, ao contrário das aplicações anteriores monolíticas, estando cada uma com suas próprias bases de dados e definições.
- **1975-1985: gestão da base de dados,** a remoção da gestão dos dados da aplicação, aumentando em grande escala o aparecimento de DMBS.
- **1985-1995: interface do utilizador**, a remoção da gestão da interface das aplicações.

- **1995-2005: *workflow***, a retirada dos processos de negócios das aplicações.

Face a evolução de sistemas de informação nas últimas décadas, sobretudo com a retirada dos processos de negócios de dentro das aplicações, [38] faz uma analogia, comparando WfMS a DMBS, no sentido que assim como as bases de dados são desenvolvidos e usados com o auxílio de DMBS, também os WfMS podem ser usados para definir e usar sistemas de *workflows*.

O termo *workflow*, que é o foco da WfMS, é frequentemente colocado na relação de dependência entre a gestão e análise de processos, podendo também ser entendido, como a forma como o trabalho flui através do processo, as atividades que são realizadas sobre ele, as pessoas envolvidas e as informações necessárias para a sua conclusão.

É comum encontrar referências a gestão de processos como sendo gestão de *workflows* e sistemas de informação que suportam gestão de *workflows* como sendo chamados de WfMS [16]. O termo gestão de *workflows* é muitas vezes referido como sendo o controlador das ações tomadas em documentos movendo-se através de um processo de negócios. Especificamente, WfMS é usado para determinar e controlar quem pode aceder a que documento, que operações os funcionários podem executar em um determinado documento e a sequência de operações que são executadas [16], sendo capaz de fazer a integração entre um conjunto de sistemas. Da mesma forma, estes também podem ser usados para implementar processos de negócios em que pessoas estão ativamente envolvidos, melhorando assim a colaboração entre os intervenientes [19], [18] e a sistematização, no sentido em que todos os trabalhadores poderão desempenhar uma tarefa seguindo os mesmos passos previamente modelados.

Face às novas necessidades, este tipo de sistemas (WfMS), foi e está sujeito a constantes inserções de novas funcionalidades, tornando-o flexível, para que possa lidar com a crescente complexidade dos processos modernos. Decorrente dessa abrangência e a expansão da flexibilidade, este acabou por evoluir incorporando mais capacidades, deixando de estar focado apenas na automatização de processos. Nos tempos atuais é comum encontrar referências a WfMS como sendo Sistemas de Gestão de Processos de Negócios (Business Process Management Systems - BPMS). WfMS é entendido como sendo o embrião dos BPMS. É importante igualmente realçar que atualmente o termo WfMS é pouco utilizado acabando mesmo por cair em desuso face a BPMS que é a evolução natural do mesmo.

Desta forma, depois da definição e de uma retrospetiva histórica, já é possível compreender um pouco mais sobre o termo *workflow* e as tecnologias associadas, sendo agora pertinente apresentar alguns conceitos chaves e as referidas descrições, definidas pela WfMC (Tabela 2).

Tabela 2: Conceitos básicos sobre workflows definidos pela WfMC [39].

Conceitos	**Descrição**
Atividade	Conjunto de eventos que ocorrem sob a responsabilidade de um ator.
Deadline	Assente no tempo de espera que cada regra de uma atividade ou lista de tarefas precisa para completar seus objetivos.

Evento	Ocorrência de uma condição causando uma ou mais ações dentro do sistema.
Gatilho	Permite o despoletar de uma atividade a partir de um evento.
Instância	Representa a associação ou inclusão de dados a uma determinada fase de um processo ou atividade
Lista de tarefas	Associada a um participante, ou grupo de participantes de um *workflow*, que podem partilhar de uma lista comum.
Papel	Atores que possuem qualificações e direitos que os tornem aptos à execução de uma atividade.
Participante	Recurso que executa o trabalho representado por uma instância de atividade, podendo ser humano, equipamentos, regras ou uma unidade organizacional.
Processo	Conjunto de procedimentos ou atividades relacionadas com o objetivo de alcançar um objetivo dentro de um contexto organizacional.
Sincronismo	Representação formal da interação das atividades, estabelecendo a dependência entre elas e específica quais tarefas devem ser executadas em paralelo, ou quais podem ser adiados até a conclusão de outra atividade.
Tarefa	Representa o trabalho realizado por um participante numa instância de processo dentro do contexto de *workflows*.
Transição	Passagem de um estado para o outro.

2.4.2 Sistemas de *workflow* integrados e Sistemas de *workflow* independentes da aplicação.

Tradicionalmente, os sistemas são desenhados e implementados não apenas por códigos procedimentais, mas também pela lógica do processo realizada pela aplicação. Ter um sistema onde a lógica do negócio é implementada usando códigos procedimentais ou funcionais, traz consequências, onde face a crescente complexidade e demanda para adaptá-las a novos requisitos, qualquer modificação do processo requer modificação do código. Neste caso, o código necessitará de ser modificado, testado e mantido, pelo que cada modificação consome recursos consideráveis. Desta forma utilizar sistemas de *workflow* facilita a tarefa de modificação da lógica de negócio, dado que se consegue percecionar visualmente o impacto das alterações e não há necessidade de alteração de código funcional.

Assim sendo, cada vez mais os sistemas de *software* integram um componente de *workflow* que facilita a modelação de processos de negócios. Esses componentes não são sistemas autónomos, são incorporados na aplicação, sendo conhecidos por Sistemas de *workflow* integrados (Figura 3), que consistem em atividades e na sua ordenação causal e temporal, realizadas por um sistema comum [19], [18].

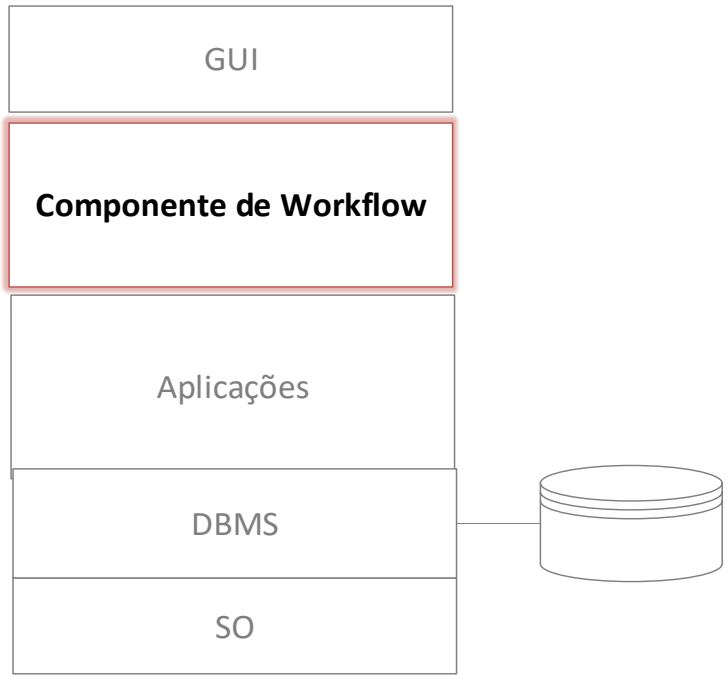


Figura 3: Arquitetura conceptual de Sistemas de workflow integrados [18], [19].

Os sistemas de *workflow* integrados contêm um componente de *workflow* dedicado que é alimentado com modelos visuais que capturam a lógica do processo, bem como informações de execução, utilizando as funções realizadas pela aplicação.

No caso da existência de mais do que um sistema, surge o designado Sistemas de *workflow* independente da aplicação (Figura 4), é nada mais do que WfMS dedicados, que garantem a invocação das aplicações conforme especificado no modelo de processo, bem como a transferência de dados entre esses sistemas [18], [19]. Este interage com outras aplicações tais como ERP, gestor de inventários entre outros, desempenhando um papel de "driver" na troca de informação entre eles, gerindo toda a lógica de dentro das organizações.

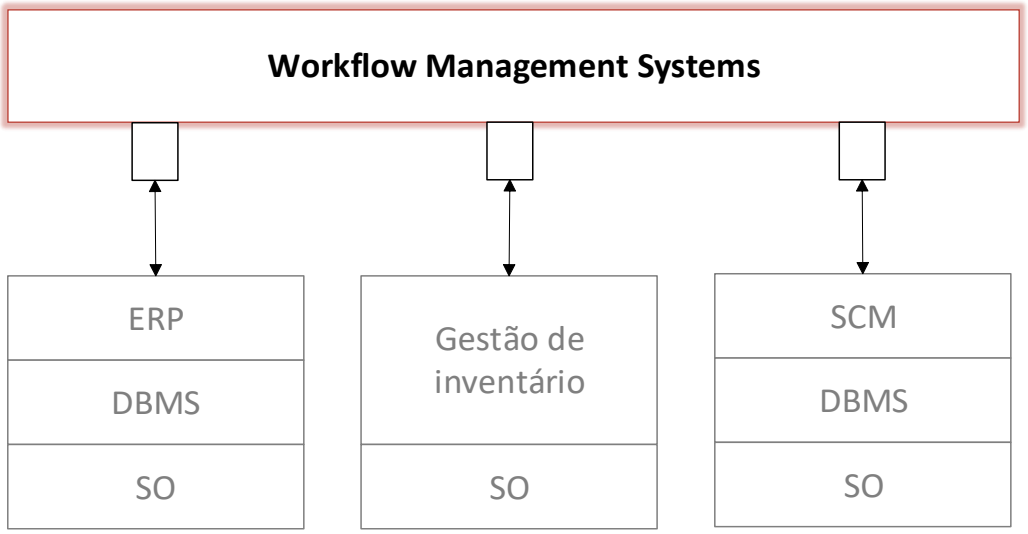


Figura 4: Arquitetura conceptual de Sistemas de workflow independente da aplicação [18], [19].

2.4.3 As três áreas funcionais de sistemas de *workflow*

As três áreas funcionais de sistemas de *workflow*, foram propostos pelo modelo de referência da WfMC, que como referido, descreve um modelo comum para a construção de sistemas de *workflow*, identificando o relacionamento entre as várias abordagens alternativas de implementação [39]:

- Funções de tempo de construção
- Funções de controlo de tempo de execução
- Interações em tempo de execução

Estas áreas podem ser divididas em duas secções (construção e execução, Figura 5), sendo que o tempo de construção comporta operações como o desenho de processos (dando origem a um modelo visual) e definições de processos. Estas definições são posteriormente utilizadas pelo *Workflow Enactment Service* para efetuar as suas operações de instanciação e controlo.

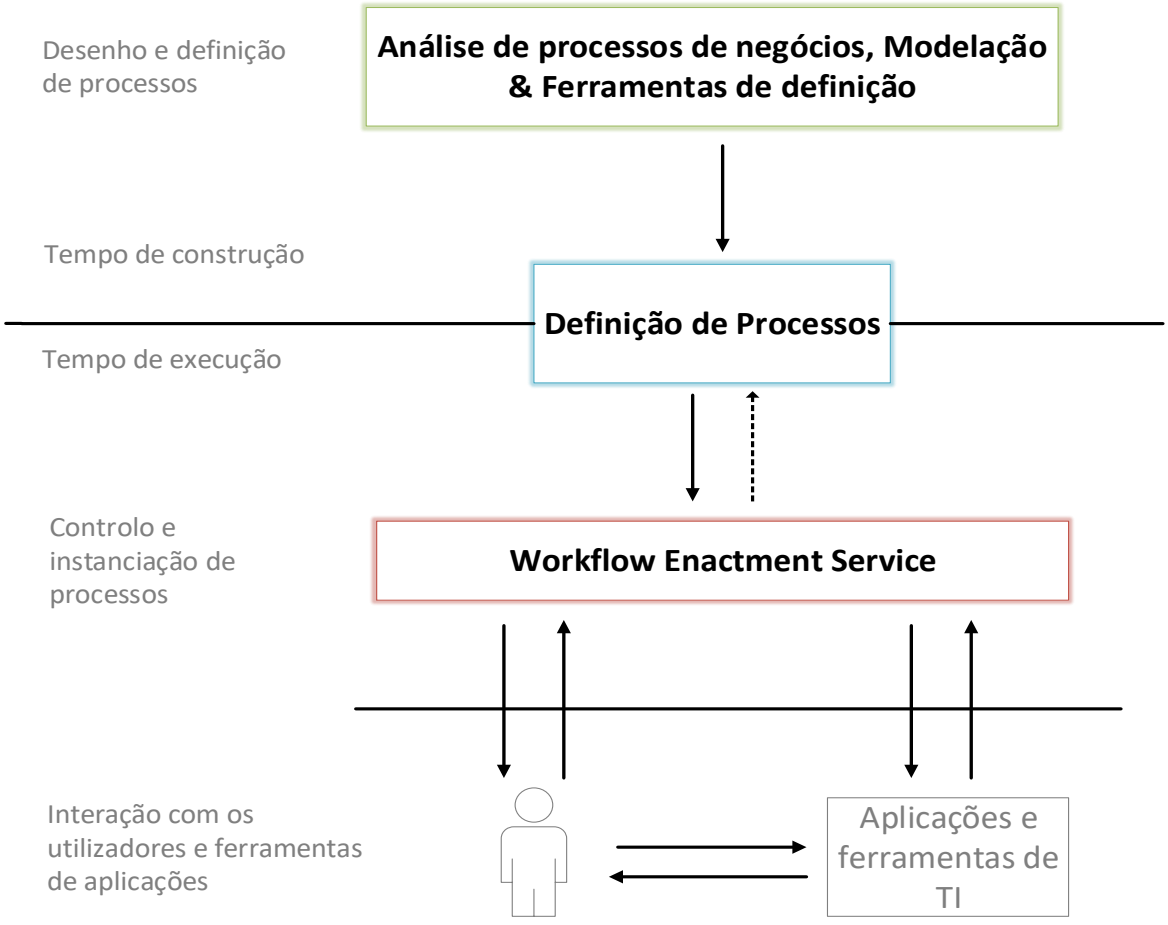


Figura 5: Características de sistemas de workflow na construção e execução [32].

Na Tabela 3, constam as descrições das ocorrências nestas três principais áreas funcionais de um sistema de *workflows*. Englobam desde o modelo de processo aos recursos humanos nas organizações que alavancam este tipo de sistemas.

Tabela 3: Áreas funcionais de WfMS [39].

Áreas funcionais	**Descrição**
Funções de tempo de construção	Durante esta fase, um processo de negócios é traduzido do mundo real para uma definição formal (designado de modelo de processo, metadados de processo ou uma definição de processo), que seja compreendido por computador, pelo uso de uma ou mais técnicas de análise, modelação e definição de sistema.
Funções de controlo de tempo de execução	Em tempo de execução, a definição do processo é interpretada pelo motor de *workflows*, componente principal de uma WfMS, responsável por criar e controlar instâncias do processo, agendando as atividades dentro do processo e invocando os recursos de RH e de TI apropriados.
Interações em tempo de execução	A interação com o motor de *workflows* é necessária para a transferência do controlo entre atividades, verificar o estado das operações nos processos, invocar ferramentas e transferir os dados apropriados para o sistema de processamento.

2.4.4 Workflow Enactment Service

Workflow Enactment Service é o centro de todas as interações na automatização de processos. Normalmente é um serviço que consiste em um ou mais motores de *workflow*, para criar, gerir e executar instâncias de *workflows*. Este pode ser considerado como uma máquina de transição de estados, onde instâncias de processo ou de atividades individuais modificam estados em resposta a eventos externos ou a decisões de controlos específicos tomadas por um motor de *workflow* [39]. Os estados de execução destas transições nas instâncias de processos têm diferentes designações (Figura 6).

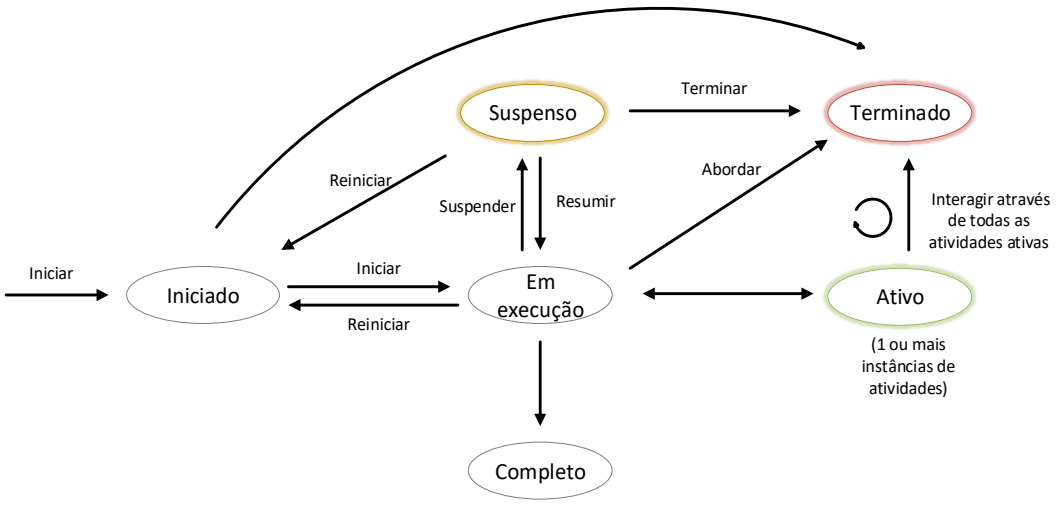


Figura 6: Exemplo de estados de execução para instâncias de processos nas transições de estados [39].

A transição deve ocorrer tanto em resposta a eventos, ou também como resultado de condições de transição dentro da definição de processo, sendo possível estarem nos seguintes estados de execução (Tabela 4) [39]:

Tabela 4: Possíveis estados de execução de uma instância de processos [39].

Estado de execução	**Descrição**
Iniciado	Cria-se uma instância de processo, incluindo propriedades do estado de processo e dados relevantes do *workflow*, mas estando ainda sem cumprir as condições para que inicie a execução.
Em execução	A instância de processo iniciou a execução e qualquer uma das suas atividades pode ser iniciada.
Ativo	Uma ou mais atividades do fluxo foi iniciada.
Suspenso	A instância de processo é pausada e nenhuma atividade é iniciada até que o processo tenha retornado para o estado em execução.
Completo	A instância de processo preenche as condições para a conclusão, podendo a instância ser destruída.
Terminado	A execução da instância de processo foi interrompida antes de sua conclusão normal, podendo a instância ser destruída

Por vezes, há a necessidade de suspender ou encerrar determinada atividade em execução dentro de uma instância de processo. Tal pode não ser possível, visto que pode haver tarefas ainda por realizar dentro da atividade. Assim sendo, só depois da atividade terminar as tarefas e já com a instância do processo retornada para um estado em execução [39], será possível suspender ou encerrar uma determinada atividade.

Contudo, a WfMC para lidar com este problema, criou padrões de transições de estados de execução exclusivos para as instâncias de atividades, diferentes das instâncias de processos (Figura 7) (Tabela 5). Assim sendo, se houver a necessidade de suspender ou encerrar uma atividade, basta ativar um dos estados de execução para as atividades. Assim, a execução de *workflow* pode ter informação sobre o estado de execução para as instâncias dos processos bem como para as instâncias de atividades apesar das atividades estarem dentro de um processo.

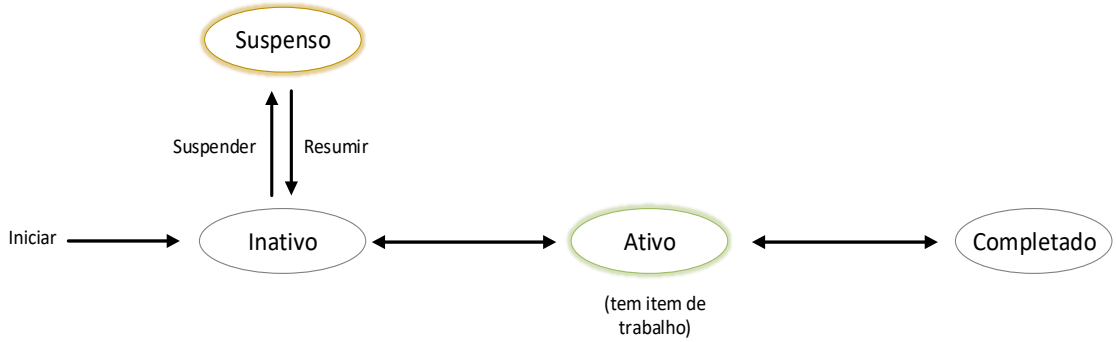


Figura 7: Exemplo de estados de execução para instâncias de atividades nas transições de estados [39].

Tabela 5: Possíveis estados de execução de uma instância de atividades [39].

Estado de execução	**Descrição**
Inativo	A atividade dentro da instância foi criada, mas não foi ativada não tendo nenhum item para processar
Ativo	Foi criado um item e atribuído à instância de atividade para processamento.
Suspenso	A instância foi pausada e não será alocado um item para processamento, até retornar ao estado em execução.
Completo	A execução da instância foi concluída.

2.5 Modelo de referência proposto pela WfMC e a evolução

Após a apresentação dos conceitos nas subsecções anteriores, importa discutir e apresentar o modelo de referência proposto pela WfMC e a evolução decorrente desta norma. Este modelo assenta numa proposta de arquitetura e normas suportada por um conjunto de componentes.

Desta forma, pretende-se abordar nesta secção a arquitetura, os componentes, a tipologia bem como a diferenciação de dois conceitos que devido a evolução dos sistemas de *workflow* tem sido muitas vezes utilizado de forma errónea (WfMS e BPMS).

2.5.1 Componentes de WfMS

O modelo de referência proposto pela WfMC, tornou-se bem estabelecida no mundo da automatização de processos, assenta em seis componentes [39]: a ferramenta de definição de processos, a ferramenta de administração e monitorização, aplicações para clientes do *workflow*, aplicações invocadas, outros *workflow enactment services* e pela API de *workflow* e formatos de intercâmbio (*Workflow Enactment Services*).

A Figura 8 apresenta a arquitetura proposta pelo modelo de referência supramencionado. Esta encontra-se dividida em componentes designados de interfaces. As interações entre as interfaces são numeradas de 1 a 5. A interface 1 refere à interação entre o motor e as ferramentas de definição de processo, enquanto que a interface 2 refere à interação entre o motor e o manipulador da lista de trabalho [40] e por fim a interface 5 que refere à interação entre o motor e as ferramentas de administração e monitorização [37], [39].

Face aos novos paradigmas de desenvolvimento de *software* com o intuito de melhorar, fez com que fosse suprida a interface 3 que geria a interação entre as aplicações e o motor de *workflows*, muito por culpa do surgimento de Web-Services. Da mesma forma, a interface 4 que abordava a integração com serviços também foi suprida, uma vez que isso pode ser realizado através de interfaces de Web-Services [37].

Sendo assim, acabou-se por permanecer apenas três dessas interfaces (interface 1, 2 e a interface 5), que anteriormente faziam parte de ferramentas focadas na automatização de processos (WfMS), não abrangendo as funcionalidades de inteligência de processos e tendo pouco suporte a modelação de processos, que acabaram por evoluir para BPMS [37].

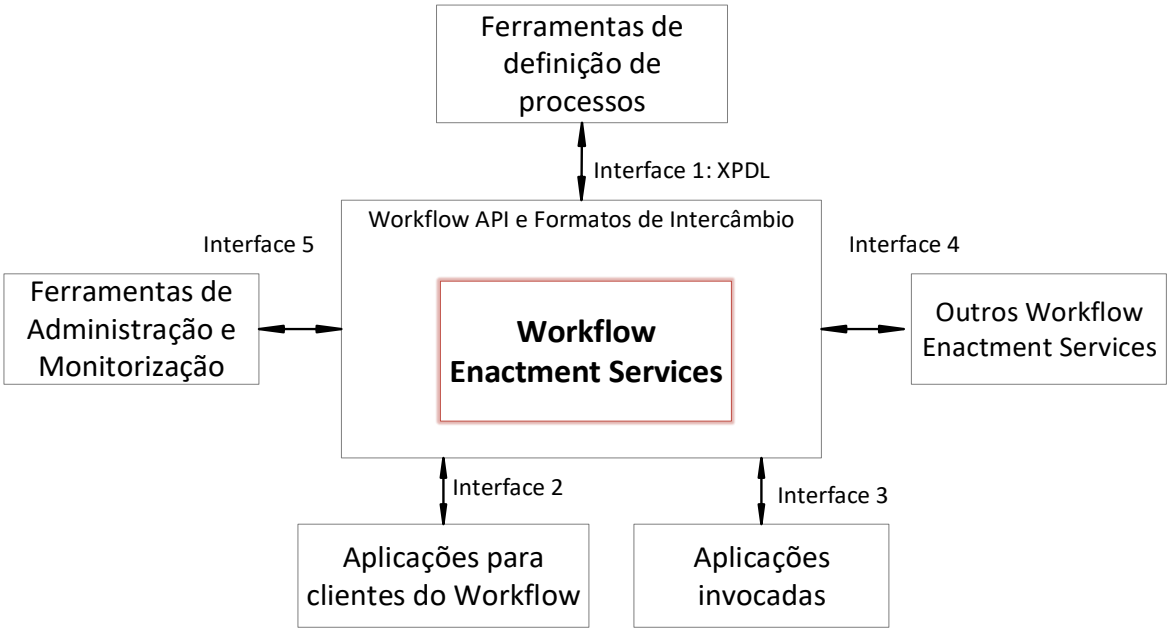


Figura 8: Componentes e interface de workflows, proposto pela WfMC [18], [19], [39], [41], [40].

Nos finais de 90 e início de 2000 depois da padronização feita pelo WfMC, quando se referia a WfMS, entendia-se como sistemas que tinham única e exclusivamente o enfoco na automatização de processos, mas com o passar dos anos e a evolução, este quase que não se usa, mas sim, o termo BPMS para generalizar sistemas que têm como principal objetivo a automatização de processos.

Assim sendo, abordando a arquitetura de BPMS, depara-se que é composta pelo motor de *workflows*, pela ferramenta de modelação de processos, pelo manipulador da lista de trabalho e pelas ferramentas de administração e monitorização (Figura 9), sendo componentes chaves.

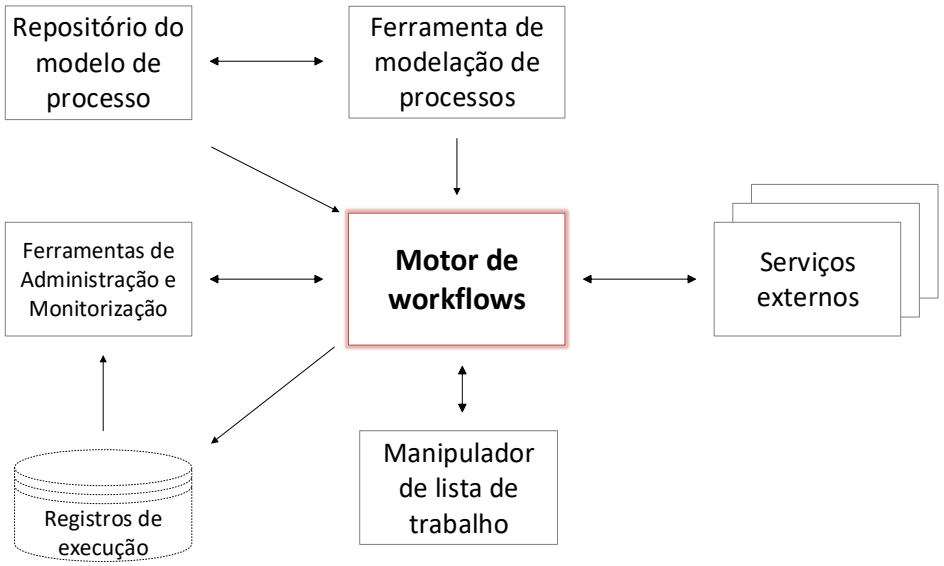


Figura 9: Arquitetura de BPMS [18], [19], [37].

A respeito do motor de *workflows*, este fornece diferentes funcionalidades, desde a capacidade de criar instâncias de processo, a capacidade de distribuir o trabalho aos participantes do processo, a capacidade de recuperar e armazenar automaticamente os dados necessários para a execução do processo e de delegar atividades interagindo com os demais componentes tal como a API de *workflow* e formatos de intercâmbio do modelo de referência de WfMC.

Por outro lado, a ferramenta de modelação de processos oferece a capacidade aos analistas de TI e analistas de negócio de criar e modificar modelos visuais, a capacidade de anotar modelos de processo com dados adicionais, regras de negócios associadas a atividades ou medidas de desempenhos associadas a um processo ou uma atividade, bem como a capacidade de armazenar, compartilhar e recuperar modelos de processo a partir de um repositório [18], [19], [37], equivalente a ferramenta de definição de processos na referência WfMC.

O manipulador da lista de trabalho permite aos participantes do processo ter acesso aos itens de trabalho enquanto que as ferramentas de administração e monitorização são necessárias na administração de todas as questões operacionais de um BPMS. Por último não menos importante, tem-se os serviços externos que devem ser capazes de expor uma interface de serviço [37].

Esta arquitetura genérica é atualmente a base para o funcionamento de qualquer tipo de sistemas que assenta na automatização de processos, sendo a evolução do modelo de referência proposto pela WfMC.

2.5.2 Diferenças entre BPMS e WfMS

BPMS pode ser entendido como sendo a classe de *software* que permite às organizações conceberem soluções de TI centradas no processo [15], capazes de integrar pessoas, sistemas e dados, preenchendo a lacuna existente entre TI e o negócio, permitindo o envolvimento próximo dos intervenientes na conceção da solução [16]. Este pode ser adaptado a processos específicos de qualquer tipo, com a finalidade de coordenar um processo automatizado do negócio de tal maneira que todo o trabalho é feito no momento adequado pelo recurso certo [42].

É comum haver equívocos relacionados a BPMS e WfMS. Estes equívocos acontecem em muitos desenvolvimentos no mundo tecnológico, onde uma nova tecnologia, mesmo que seja evolutiva, é comercializada com um novo nome, em consequência do marketing e em parte para diferenciar-se dos existentes [43]. Como exemplo tem-se os primeiros BPMS que focaram em *workflows*, sendo atualmente considerados uma evolução de WfMS. BPMS é conhecido como uma abordagem que se concentra na captura e melhoria dos processos de negócios para tornar uma organização mais eficiente, acabando por considerar *workflows* como sendo apenas relacionado a modelação ou talvez a execução de processos [43]. Por outro lado, WfMS é uma ferramenta que permite configurar e monitorizar uma sequência de tarefas definida, sob a forma de um diagrama de *workflows*, onde especialistas descrevem-no como sendo Business Process Management Lite. Este tende a focar-se mais nas pessoas que executam as tarefas do que nos próprios processos, ao contrário de uma BPMS. WfMS é apenas uma de muitas tecnologias encontradas em um BPMS, sendo BPMS geralmente composto por definição e modelação de processos; automatização de *workflow*; análise e gestão de processos; otimização de processos; EI; monitorização de atividades; EAI, tal como ilustrado na Figura 10.

Figura 10: Modelo conceptual dos componentes de BPMS.

Face às descrições supra, é possível constatar como diferenças chaves:

- A capacidade que os WfMS têm de coordenar as interações entre pessoas e sistemas de software, enquanto que os BPMS coordenam interações entre todos os recursos em uma organização;
- A maior complexidade dos BPMS, apresentando interações em todos os departamentos e recursos externos, além de serem mais comumente usados para auxiliar as organizações na implementação de melhoria contínua, contrasta com WfMS que pode ser usado apenas para modelar e automatizar etapas num processo específico.

2.6 SOA e integração de serviços a processos

Neste estudo é pertinente fazer referência às arquiteturas orientadas a serviços (SOA), tendo este trazido ganhos significativos para a automatização de processos. Segundo a definição oficial da OASIS [44], é um paradigma para organizar e utilizar recursos distribuídos que podem estar sob o controle de diferentes domínios de propriedade.

Apesar da definição oficial poder ser ambígua, é possível definir SOA como sendo um paradigma para conceber sistemas de *software* a partir de componentes reutilizáveis designados por serviços [20], sendo serviço um conjunto de blocos de códigos que desempenham funções distintas [6], [23], [45], [46]. O objetivo do desenvolvimento baseado em SOA é permitir às organizações montar sistemas de negócio a partir de módulos simples, para obtenção de flexibilidade, agilidade e melhorias de produtividade no desenvolvimento das suas TI [6], capazes de relacionar os processos de negócio com os processos técnicos [47] e fazer o mapeamento das relações de *workflows* entre ambos [23].

SOA normalmente assenta no princípio de disponibilizar serviços de negócios reutilizáveis capazes de serem montados a partir de subcomponentes de *software* de tal forma que o fornecedor (o que disponibiliza o serviço)

e o cliente (o que o utiliza) são considerados fracamente acoplados (definição de interface neutra que não está fortemente ligada a uma implementação particular) [6], [23], [46]. Estes não precisam de ter um conhecimento aprofundado um do outro, desde as tecnologias, plataformas [48], localização ou ambientes que os rodeia, podendo o código ser reutilizado graças às suas características de interface neutra, facilitando a tarefa dos programadores que precisam de apenas descobrir as interfaces das aplicações já desenvolvidas, ao invés de construir novos sistemas a partir do zero, sempre que novas regras de negócios forem alteradas ou adicionadas [13].

Os serviços em SOA possuem componentes que se relacionam através de interfaces e contratos bem definidos. A interface é definida de forma neutra e independente do *hardware*, do sistema operativo e da linguagem de programação [47] em que é implementado, permitindo que estes serviços interajam uns com os outros de maneira uniforme e universal.

A possibilidade que a arquitetura disponibiliza de ter-se serviços fracamente acoplados, permite que serviços fornecedores com estas características, beneficiem da sua agilidade e habilidade para sobreviver às mudanças constantes na estrutura e na implementação interna que compõe a aplicação [23]. Além de ter os serviços como componente chave da arquitetura, SOA também é constituído por interfaces de serviços que possibilitam a interação entre o fornecedor e o cliente.

Embora não haja um padrão oficial para a composição de SOA, a comunidade concorda com quatro princípios fundamentais como princípios orientadores para alcançar SOA [49], [50], [47], sendo eles, os limites do serviço devem ser explícitos; os serviços devem ser autónomos; os clientes e serviços devem partilhar contratos e não códigos e por fim a possibilidade de compatibilidades baseada na política. A compatibilidade baseada na política entende que dois requisitos ortogonais devem ser suportados numa interação com um serviço. A funcionalidade, sintaxe e semântica do fornecedor deve atender aos requisitos do consumidor; e as capacidades e necessidades técnicas devem coincidir. Contudo, os aspetos funcionais são descritos na interface de serviço, mas as capacidades, necessidades ortogonais e não funcionais são especificadas usando políticas [51].

A arquitetura SOA é também conhecido como uma evolução da arquitetura cliente/servidor, onde a lógica de negócio é decomposta a níveis menores de granularidade, sendo baseadas nas ideias e tecnologias implementadas em XML e WS[23]. WS é conhecido como um sistema de *software* desenhado para suportar interações interoperáveis máquina a máquina, tendo como elementos-chave [6], [46]:

- Web Services Definition Language (WSDL), padrão para representar partes de *software*;
- Simple Object Access Protocol (SOAP), protocolo para troca de informações em um ambiente descentralizado e distribuído;
- Universal Description, Discovery, and Integration (UDDI), mecanismo de registo de serviço, baseado em XML. Permite que serviços sejam descobertos na Web e tornem interoperáveis. É comparado às páginas amarelas de uma lista telefônica, onde as empresas podem-se listar por nome, produto, localização ou os serviços que oferecem.

Desta forma, a comunicação com WS é feita a partir da rede usando tecnologias como XML (relacionada a fundamentos de dados e protocolo), WSDL (relacionada a descrição do serviço), SOAP e UDDI (Figura 11),

sendo esses capazes de fornecerem uma abordagem comum para definir, publicar e usar WS acessíveis através de um protocolo padrão, especificamente SOAP [45]. Um WS normalmente permite a descoberta do serviço, o cliente faz uma solicitação ao fornecedor, este processa a solicitação e envia a resposta para o cliente [6], [46].

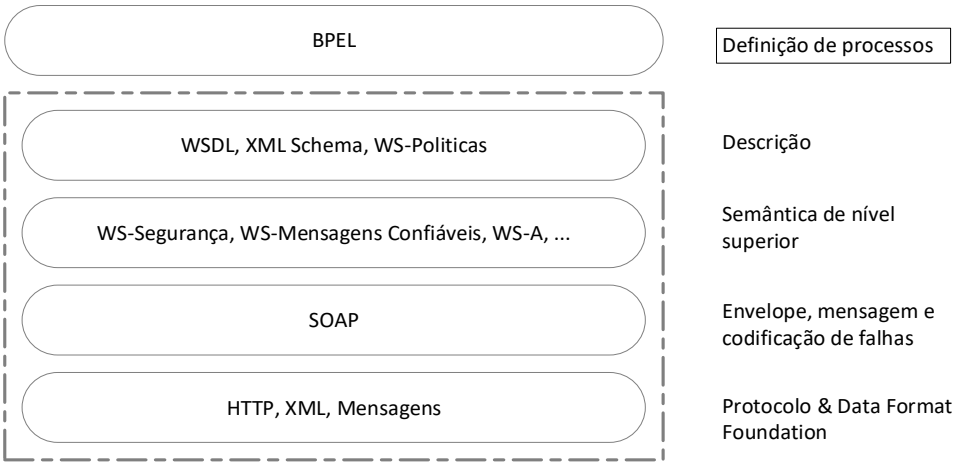


Figura 11: Padrões em torno de WS mais a integração de serviços a processos [23].

Paralelo à utilização de SOAP, o WS também permite a utilização de Representational State Transfer (REST). REST é um estilo arquitetural que usa tipicamente HTTP/HTTPS, enquanto SOAP é independente da camada de rede podendo usar uma variedade de protocolos de rede.

Pode-se dizer que WS fornecem a tecnologia base para implementar a arquitetura SOA, mas não é a única. Common Object Request Broker Architecture (CORBA- Arquitetura e infraestrutura independentes de fornecedores que permite sistemas comunicarem numa rede [52]) e sistemas Message-Oriented Middleware (MOM - classe específica de middleware que suporta a troca de mensagens de propósito geral em ambientes de sistemas distribuídos [53]), também podem ser usados. SOA não é específico da linguagem, podendo os serviços serem implementados em qualquer linguagem de programação que permita gerar e interagir com WSDL [23]. Desta forma a utilização de SOA permite obter ganhos circunstanciais na organização e no desenvolvimento de *software* (Tabela 6):

Tabela 6: Benefícios de SOA [23], [20].

Benefícios	**Descrição**
Centrado em negócio	Desenvolvimento de *software* centrada no negócio.
Menor custo a longo prazo	Menor custo incremental para o desenvolvimento.
Skills reduzido	Menor exigência quanto a treinamento e conhecimentos técnicos.
Custos baixos	Diminuição dos custos no desenvolvimento de *software*.
Rapidez	Ciclo de desenvolvimento de *software* mais rápido.

Independente da plataforma	Ajuda na integração de aplicações entre várias partes, pode alavancar sistemas legados e criar funcionalidades adicionais sem ter que reconstruir todo o sistema.
Focused developer roles	Os programadores podem focar-se completamente na implementação e manutenção dos seus serviços sem ter que se preocupar com serviços de outros, desde que o contrato predefinido seja respeitado.
Independentes da localização	Independentemente da localização é possível ao consumidor do serviço procurar e descobri-lo.
Reutilização de código	A divisão das aplicações em pequenas partes destintas, ajuda na reutilização em vários outros sistemas, reduzindo assim o custo do desenvolvimento.
Maior testabilidade	Serviços pequenos e independentes são mais fáceis de testar e depurar do que aplicações monolíticas, permitindo ter *softwares* mais confiáveis.
Desenvolvimento paralelo	A independência entre os serviços com contratos predefinidos, ajuda o desenvolvimento em paralelo reduzindo consideravelmente o ciclo de vida do desenvolvimento de *software*.
Melhor escalabilidade	Possibilidade ter várias instâncias do serviço em execução em servidores diferentes, sendo possível mover o serviço de forma transparente para servidores com mais recursos.
Maior disponibilidade	A possibilidade de ter várias instâncias de um serviço e não importar com localização do mesmo, é possível garantir a alta disponibilidade.

2.7 Mecanismos de segurança

Outro ponto que se torna importante abordar na realização deste trabalho tem a ver com os mecanismos de segurança. Dentro do universo dos mecanismos de segurança, tem destaque para este trabalho a forma de acesso a informação ou recursos "em nome de"; ou seja, permitindo que terceiros (uma determinada aplicação ou serviço) comuniquem com outras aplicações ou serviços, agindo em nome do utilizador, e não enquanto a aplicação ou serviço.

A autorização a terceiros utilizando o modelo tradicional, onde o cliente acede a um recurso protegido, usando as credenciais do proprietário do recurso, tem associado a si vários problemas e limitações, uma vez que o proprietário terá de partilhar suas credencias com terceiros [54]:

- As credenciais precisam ser armazenadas em aplicações de terceiros para uso futuro.
- Serão necessários servidores para suportar a autenticação.
- As aplicações de terceiros ganham acesso a todos os recursos protegidos do proprietário do recurso, deixando-os sem capacidade de restringir a duração ou acesso a um subconjunto limitado de recursos.

- O proprietário do recurso não consegue revogar o acesso a terceiro sem revogar o acesso a todos os outros, só sendo possível com alteração da palavra passe.
- A quebra de segurança na aplicação terceira, resulta na exposição das credenciais e de todos os dados protegidos nesta.

Para mitigar estes problemas, destacam-se dois mecanismos, OAuth [54] e acesso baseado em *tokens* [55]. O OAuth é um padrão aberto que permite o acesso autorizado de clientes a recursos disponibilizados por um servidor. O padrão OAuth também fornece métodos para os clientes autorizarem o acesso, por parte de terceiros, a recursos e informação de que são titulares, e que é disponibilizada por um servidor sem ser necessário inserir ou partilhar as credenciais de acesso desse utilizador.

Os problemas e limitações identificados no modelo tradicional de autorização são tratados por este padrão com a introdução da camada de autorização, separando o papel do cliente, do proprietário do recurso. O cliente solicita o acesso a recursos controlados pelo proprietário do recurso, hospedado no servidor de recursos, sendo emitidas credenciais diferentes das do proprietário do recurso. Em vez de usar as credenciais do proprietário do recurso para aceder a recursos protegidos, o cliente obtém um *token* de acesso que indica um âmbito específico, tempo útil e outros atributos de acesso, sendo esses *tokens* emitidos por um servidor de autorização com a aprovação do proprietário do recurso. O cliente usa o *token* de acesso para aceder aos recursos protegidos hospedados pelo servidor de recursos (Figura 12) [54].

Figura 12: Fluxo de autorização OAuth [54].

Por outro lado, a autorização baseada em *Tokens* é menos complexa e ao contrário do OAuth não é um padrão. O cliente acede ao servidor enviando as credenciais de acesso, o fornecedor do serviço valida as credenciais, guardando um *token* de autorização em caso de sucesso, associando o *token* ao endereço do cliente e atribui o tempo de vida deste *token* de acesso. De seguida, o serviço fornecedor envia uma mensagem de sucesso para o cliente, juntamente com o *token*. O cliente passa a invocar os serviços no servidor enviando o *token* de autorização, onde o servidor irá validar cada requisição, verificando se o *token* se encontra válido (Figura 13).

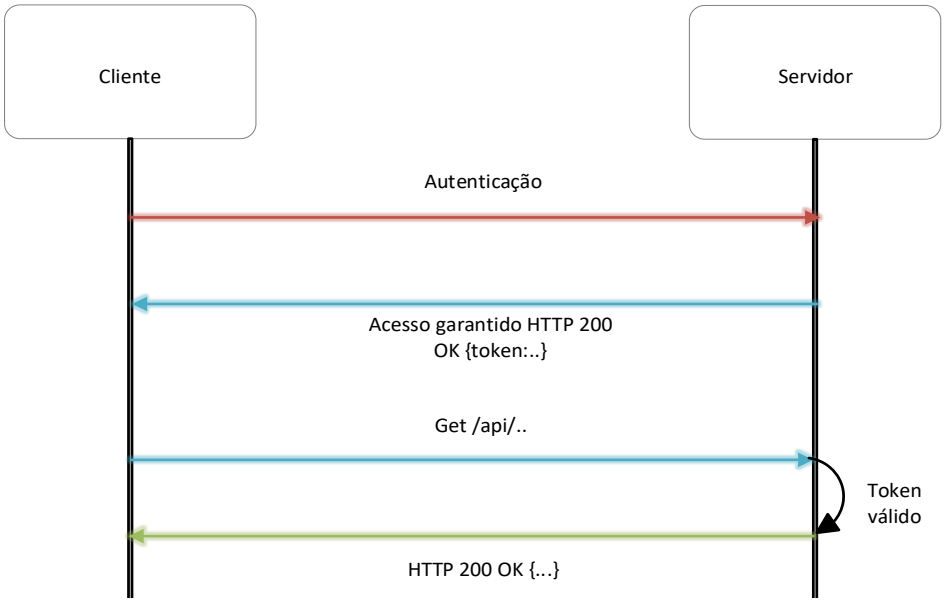


Figura 13: Fluxo de autorização baseado em tokens [55].

2.8 Resumo

No final deste capítulo, foi possível entender os conceitos básicos para a modelação de processos e conhecer as notações e técnicas mais usadas nesta disciplina. Abordou-se também os conceitos básico da automatização de processos. Esses conceitos incluem processos, atividades, tarefas, instâncias entre outros. Ainda na automatização de processos foi possível conhecer as três áreas funcionais de sistemas de *workflows*, sendo eles as funções de tempo de construção ao qual durante esta fase, um processo de negócios é traduzido do mundo real para uma definição formal. Em seguida apresentou-se as funções de controlo de tempo de execução onde a definição do processo é interpretada pelo motor de *workflows* bem como, as interações em tempo de execução tornando-se pertinente a interação com o motor de *workflows* para a transferência do controlo entre atividades.

Apresentou-se igualmente as diferentes designações dos estados de execução das transições nas instâncias de processos e nas instâncias de atividades. Nas instâncias de processos destacam-se os estados, "**iniciado"** quando se cria uma instância de processo, incluindo propriedades do estado de processo e dados relevantes do *workflow*, mas estando ainda sem cumprir as condições para que inicie a execução, "**em execução"** quando a instância de processo iniciou a execução e qualquer uma das suas atividades pode ser iniciada e "**ativo"** quando uma ou mais atividades do *workflow* foi iniciada.

Também se abordou o modelo de referência proposto pela WfMC. Onde se destacaram os componentes propostos por este padrão e fez-se a comparação com a nova arquitetura de automatização de processos. Apresentaram-se também as vantagens e diferenças entre WfMS e BPMS destacando a capacidade que os WfMS têm de coordenar as interações entre pessoas e sistemas de software, enquanto que os BPMS coordenam interações entre todos os recursos numa organização.

Para concluir o capítulo abordou-se os mecanismos de segurança para garantir a autorização a terceiros a WS, padrões e benefícios associados as arquiteturas orientados a serviços e os quatro princípios fundamentais como

princípios orientadores para alcançar SOA, sendo eles, os limites do serviço devem ser explícitos; os serviços devem ser autônomos; os clientes e serviços devem partilhar contratos e não códigos e por fim a possibilidade de compatibilidades baseada na política.

3. Abordagem em uso e problemas relacionados

Neste capítulo apresenta-se a abordagem e problemas do sistema de *workflows* ainda em uso, para nos próximos capítulos ser possível constatar as diferenças em relação a implementação da nova abordagem. Primeiramente é apresentada a tipologia dos *workflows* do SIJ, assentes em *workflows* para processo penal, processo cível, documento em versões e "requerimento e despacho".

Igualmente discute-se a arquitetura atual do sistema e os serviços auxiliares que servem de suporte a este. Seguidamente, abordam-se os principais problemas atualmente sentidos no SIJ, advindo desta abordagem, entre eles o desempenho e os recursos computacionais necessários para o funcionamento do sistema. Para concluir, apresenta-se a interface do utilizador, com as principais áreas que se encontram intimamente associadas ao sistema de *workflows*.

3.1 Tipologia de workflows do SIJ

O SIJ utiliza *workflows* para suportar a lógica de atividades e tramitação associadas a documentos e processos. Estes estão desenhados por tipo de processo, com suporte à produção de documentos, designadamente requerimentos, despachos e explicações de despachos (Figura 14).

A decisão de modelar os *workflows* por tipo de processos surge das reuniões realizados pela comissão de acompanhamento, constituída por intervenientes ligados ao sistema judicial e analistas da equipa de desenvolvimento, estando estes *workflows* divididos em partes destintas:

- Processo Penal (*workflow* referente a processo penal);
- Processo Cível (*workflows* referentes aos processos cíveis);
- Documentos em Versões (*workflow* que gere as versões de documentos);
- Requerimento e Despachos (*workflow* que gere os requerimentos e despachos);

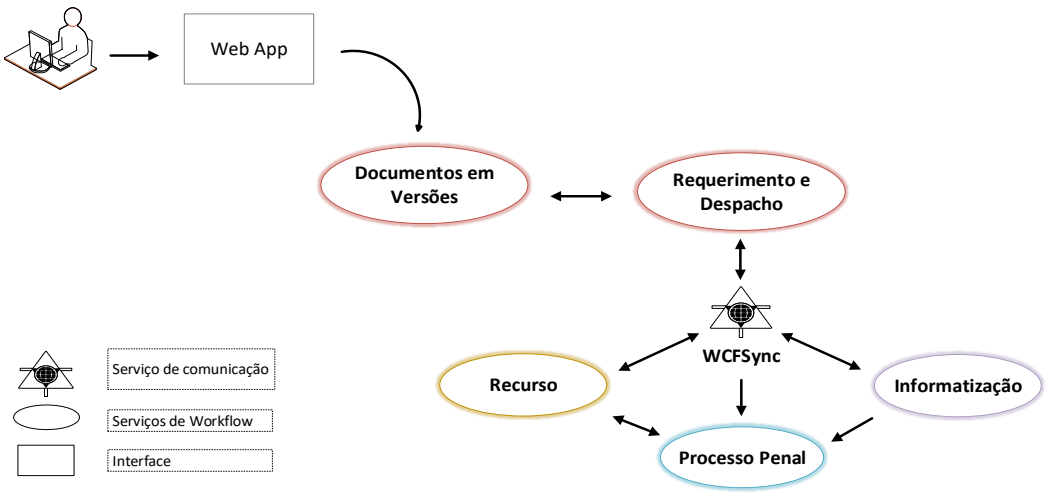


Figura 14: Sequência da execução dos workflows do SIJ [56], [57].

Os *workflows* listados anteriormente, interagem entre si, mediante uma sequência, advindo das necessidades e da lógica de negócio que assim requer. Conceptualmente, esta interação inicia-se a partir da interface web onde são efetuadas as ações pelos atores do sistema judicial, sendo esta interface adaptável mediante as funções de cada utilizador. Esta interface no despertar de um evento, comunica com os serviços de *workflows* que têm a responsabilidade de validar e efetuar tarefas dependendo da lógica modelada nos *workflows*.

Nessa interação, o primeiro *workflow* corresponde ao Documentos em Versões. Este surgiu da necessidade de controlar as versões de documentos que entram no sistema, permitindo guardar várias versões de um documento em produção, consultar as versões persistidas ou até mesmo eliminar um determinado documento em produção.

Após o documento ter sido dado como terminado, o *workflow* de Documentos em Versões interage com o *workflow* de Requerimento e Despacho. O *workflow* Requerimento e Despacho advém da necessidade de seguir um padrão normalmente utilizado no sistema judicial, referente à interposição de um pedido (requerimento), uma decisão (despacho) e o cumprimento da decisão (explicação de despacho).

Ambos os *workflows* são a base do serviço de *workflows*. A interação entre eles é feita a partir da camada de comunicação, no serviço criado exclusivamente para garantir um canal de comunicação entre as instâncias dos *workflows*. O serviço de comunicação criado exclusivamente, encarrega-se de interligar os demais tipos de *workflows* como o processo penal, recurso ou informatização.

A partir dessas explicações, é possível conhecer a sequência base da interação dos vários *workflows* que constituem o SIJ. É igualmente importante apresentar em detalhe a constituição e o funcionamento dos três principais *workflows* que são utilizados para a implementação do caso prático da restruturação em estudo (secções 3.1.3 à 3.1.3).

3.1.1 Workflow de Documentos em Versões

O *workflow* de Documentos em versões como referido na breve descrição feita anteriormente, controla as versões enquanto o documento encontra em fase de produção. Por exemplo, pode-se estar a redigir um despacho ou outro tipo de documento e não ser possível terminá-lo de imediato. A partir deste *workflow* pode-se guardar esse documento em produção e aceder posteriormente a qualquer das suas versões anteriores para edição e/ou conclusão. Este *workflow* desempenha um papel importante no que toca a interligação com os demais *workflows*. É a partir dele que se dá início à entrada de qualquer documento que se pretende inserir no sistema.

A sua lógica inicia com a submissão de um documento pelo utilizador no sistema. Com a entrada do documento o sistema avalia se este está marcado como terminado. Caso não esteja, é gravada uma versão temporária. Se estiver marcado como terminado, é gravada a versão final, e são removidas as versões temporárias (caso existam). Finalmente é invocado o próximo *workflow* em cadeia e termina a execução (Figura 15).

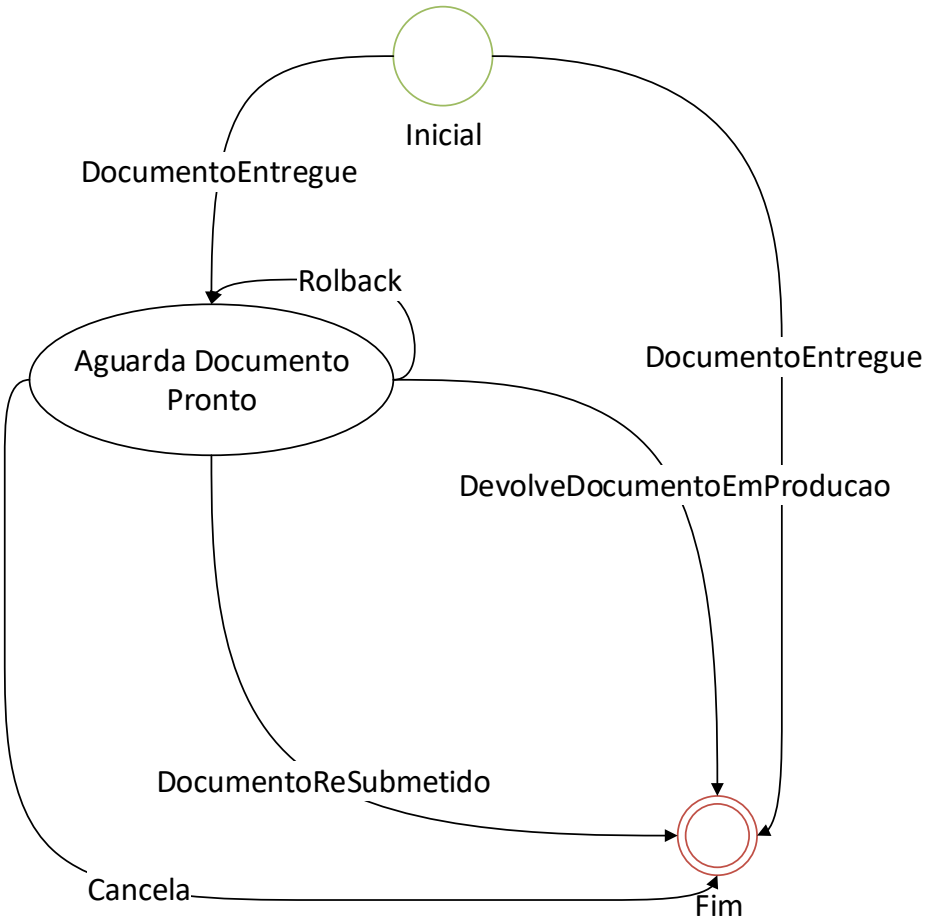


Figura 15: Modelo concetual do workflow Documentos em Versões.

3.1.2 Workflow de Requerimento e Despacho

Ao contrário do documento em versões que gere os documentos, o *workflow* de Requerimento e Despacho tem como função gerir os requerimentos e despachos que são submetidos pelos requerentes e decisores. Este possui dois caminhos de processamento: a entrada de um requerimento, ou a entrada de uma decisão, sem haver requerimento pendente. Com a entrada do requerimento, é despoletada a notificação dos intervenientes no processo que devam ser informados da existência do requerimento.

Com a entrada da decisão (podendo esta ser relativa ao requerimento, ou podendo ter dado entrada sem haver qualquer requerimento), a secretaria deve proceder à explicação da decisão. De notar que quer o requerimento, quer a decisão são blocos de texto e/ou conjuntos de documentos anexados pelo utilizador, ao nível da interface. Assim, e para que o sistema, como um todo, possa ter informação relevante (estatística, processamento de custas, medidas de coação, etc.) num formato que possa ser interpretado pelo servidor, é necessário proceder ao que se designou por explicação de despacho. A explicação de despacho é a tarefa de, caso a caso, e mediante formulário(s) próprio(s) (um por cada tipo de explicação), proceder à tradução da informação textual para informação contextualizada que possa mais tarde ser processada informaticamente. Esta tarefa é desempenhada pelos oficiais de justiça. Elaborada a explicação, são notificados os intervenientes indicados e é invocado o próximo *workflow* em cadeia (Figura 16).

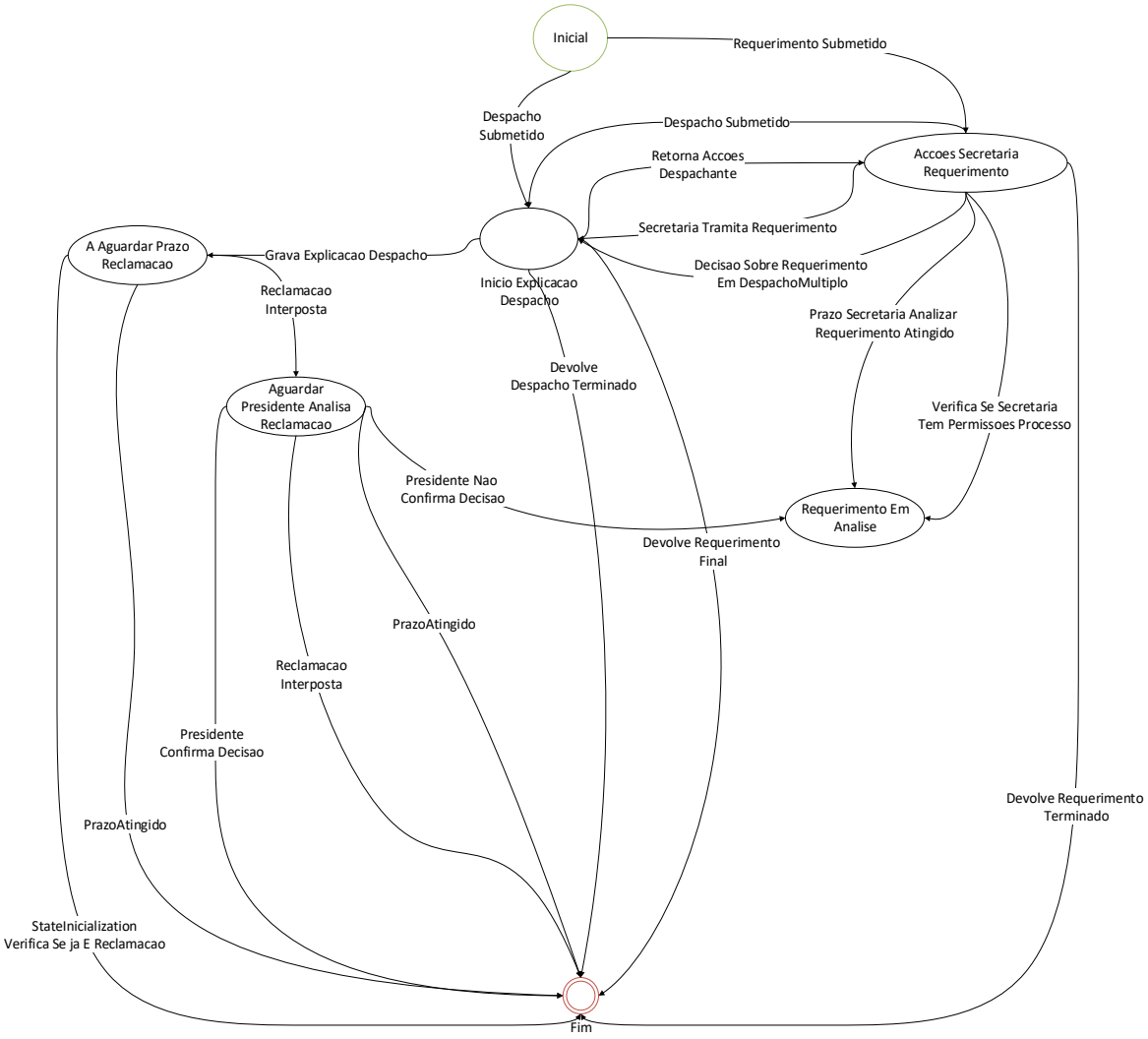


Figura 16: Modelo concetual do workflow Requerimento e Despacho.

Tal como o *workflow* de Documentos em Versões, o Requerimento e Despacho também desempenha um importante papel no sistema de *workflows*, servindo de *pivot* na comunicação com os demais, sendo possível a partir deste despoletar *workflows* referentes a processos penais ou processos cíveis.

3.1.3 Workflow de Processo Penal

Antes de entrar na descrição do funcionamento do *workflow*, é importante ter em consideração que um processo penal pode ter quatro formas diferentes, encontrando previstas no artigo 299º do C.P.P de Cabo Verde.

Segundo o código, o processo penal segue a forma comum ou especial. O processo comum segue a forma comum única que é a do processo **ordinário**. Em casos em que a lei prevê, este segue as formas especiais (artigo 300º do C.P.P), podendo ser a forma **sumário**, **transação** e o processo **abreviado** [58].

A forma de processo comum (nº 2 do artigo 299º do C.P.P) se aplica nos casos em que a lei não determine a aplicação de formas especiais, ou seja, tem carácter residual, não geral. O processo comum ordinário encontra-

se previsto nos artigos 338º e seguintes do C.P.P, cujo desenvolvimento é de aplicação subsidiária aos processos especiais em tudo quando não esteja especialmente regulada [58].

Quanto aos processos especiais, o processo sumário vem regulado nos artigos 412º e seguintes do C.P.P, e é aplicado em casos de flagrante delito em crimes com pena de prisão cujo limite máximo não seja superior a 5 anos quando à detenção tiver procedido qualquer autoridade judiciária ou entidade policial ou quando à detenção tiver sido efetuada por outra pessoa e, num prazo que não exceda a duas horas, o detido tenha sido entregue a uma autoridade judiciária ou entidade policial, tendo esta redigido auto sumário de entrega [58].

O processo de transação, de natureza negocial, vem previsto no artigo 422º e seguinte do C.P.P, para haver lugar terá de ser requerida pelo Ministério Público, pelo arguido, ou pelo assistente, bem como por aquele que tem a faculdade de se constituir assistente, que o processo siga os seus trâmites sob a forma de transação para aplicação de uma pena consensual [58].

Por último, o processo especial abreviado, vem previsto nos artigos 430º e seguintes do C.P.P., só tem aplicação quando cumpra todos os requisitos previstos nas alíneas do número 1 do mesmo artigo [58] .

Desta forma com aplicações diferentes, estas formas processuais também possuem fases processuais diferentes (Tabela 7) uma das outras. Estas diferenças, influenciam na modelação do *workflow* do processo penal, na mediada que podiam ser quatro *workflows* distintos. Mas isso não acontece, uma vez que a equipa de desenvolvimento os convergiu num só, porque apesar de terem algumas fases diferentes, no geral possuem a maioria das fases iguais, o que facilita na não replicação de modelos visuais, acabando por ficar toda a lógica num único *workflow*.

Tabela 7: Fases do processo penal.

Abreviado	**Ordinário**	**Sumário**	**Transação**
Distribuição	Distribuição	Notícia do Crime	Distribuição
Instrução	Instrução	Exame de Auto Inicial	Instrução
Aguardar Remessa do Processo para o Tribunal	Audiência Contraditória Preliminar	Distribuição	Aguardar Audiência
Aguardar Julgamento	Separação de Processo	Instrução	Audiência
Fase Julgamento	Aguardar Remessa do Processo para o Tribunal	Separação de Processo	Separação de Processo
Aguardar Trânsito em Julgado	Aguardar Julgamento	Aguardar Remessa do Processo para o Tribunal	Aguardar Remessa do Processo para o Tribunal
Transitado em Julgado	Fase Julgamento	Aguardar Julgamento	Aguardar Trânsito em Julgado

Arquivamento via MP	Aguardar Trânsito em Julgado	Fase Julgamento	Transitado em Julgado
Arquivamento	Transitado em Julgado	Aguardar Trânsito em Julgado	Arquivamento via MP
	Arquivamento via MP	Transitado em Julgado	Arquivamento
	Arquivamento	Arquivamento via MP	
		Arquivamento	

Apresentadas as fases e formas possíveis que um processo penal pode ter, sendo este influenciador na modelação do *workflow* do processo penal, resta descrever o funcionamento do *workflow* em causa. A lógica deste inicia normalmente com a inserção de um novo auto que pode ser de denúncia ou de flagrante em delito (Figura 17). Em casos de autos em flagrante, o processo é remetido para o procurador de turno, não precisando da distribuição por parte da secretaria do MP para atribuição do procurador/procuradores da secção do respetivo tipo de crime constante no processo, sendo processado como processo Sumário. Esta fase no sistema judicial é conhecida como a fase de instrução. Na fase de instrução, para além da distribuição e atribuição ao procurador de turno, o *workflow* também dá a possibilidade de serem efetuados reclamações hierárquicas, suspensão do processo, bem como poder aguardar o prazo para o pedido da abertura de ACP.

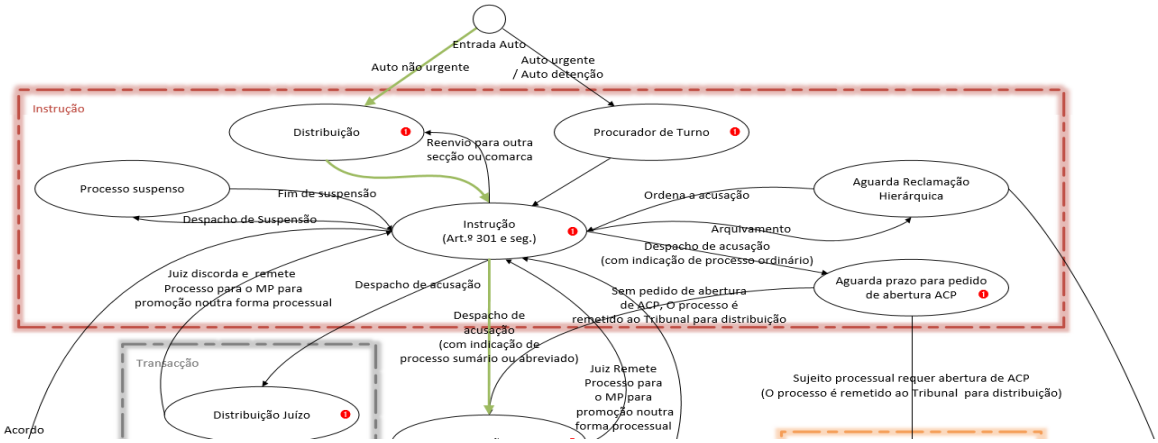


Figura 17: Modelo conceptual do workflow do processo penal.

No sistema, excetuando transições automáticas, todas as outras são despoletadas por entradas de explicações de despachos. Sendo assim para que o *workflow* avance é necessária a inserção de um despacho por parte dos magistrados ou pessoal de secretaria com competências delegadas para tal. Todas as fases que um processo pode tramitar, são previamente modeladas, levando em consideração todas as possibilidades de explicações de despachos que cada fase permite durante o ciclo de vida de um processo. Por exemplo o MP pode requerer que o processo siga na forma transação, que permite a chegada a acordo entre as partes sem prejuízo para ambos, acabando este por acontecer com a inserção de um despacho de transação para a forma transação. Assim sendo,

estando numa das fases da forma transação, é possível a distribuição ao juízo mediante o peso e o tipo de crime do processo.

Por outro lado, se o processo, ainda estiver na fase de instrução (Figura 17) e o sujeito processual requerer uma abertura de ACP (Figura 18), o *workflow* permite a tramitação para a fase de ACP e o processo é remetido para o tribunal para a distribuição do juízo. Igualmente o *workflow* permite que um processo atribuído com a forma abreviada, suprima muitas das outras fases desnecessárias para a forma de processo em causa.

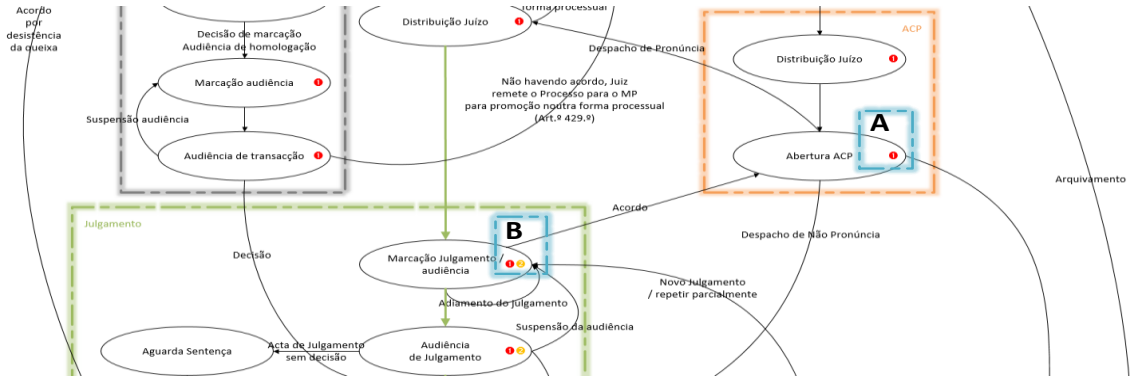


Figura 18:Modelo conceptual do workflow do processo penal, continuação, parte II.

É possível constatar que nos estados de Abertura ACP (área A Figura 18) e Marcação de Julgamento (área B da Figura 18) constam pequenos círculos associados a estes estados. Este tem a ver a eventos que não despoletam transições. Esses eventos, são muitas vezes explicações de despachos, onde no modelo visual do processo penal são representadas por pequenos círculos com cores diferentes dentro das elipses referentes aos estados. Cada um desses pequenos círculos têm associados diferentes explicações de despachos que foram agrupados mediante a ocorrência desses em mesmos estados (Tabela 8). As explicações que encontram agrupados na fase 4 da Tabela 8 acontecem em todos estados, não havendo necessidade de replicar no modelo visual, sendo apenas constantes da legenda do modelo.

Tabela 8: Fases de possíveis explicações de despachos mediante o estado que o processo se encontra no workflow do processo penal.

Fase	**Despacho**	**Fase**	**Despacho**
1	Aceitação de pedido de inquirição antecipada Alteração de rol Medidas de Coação Pedido de diligência Pedido de indeminização cível	2	Alteração de acusação Antecipação/Adiamento de audiência Contestação

Fase	Despacho	Fase	Despacho
3	Alteração de sentença Esclarecimento	4	Apensação de processos Constituição de assistente Constituição/Substituição de advogado Juncão de documentos Outra Pedido de advogado oficioso Separação de processos

Sendo assim, depois da passagem pela fase de ACP o processo pode tramitar para a fase de julgamento (Figura 18), na terminologia judicial, este diz-se que o processo já se encontra no juízo, ou seja, já foi dado como terminado pelo MP, a instrução e os factos para a acusação já foram recolhidos e a acusação efetuada, remetendo a continuação do processo para o Tribunal Judicial, para seguir os trâmites processuais. Assim, no caso de um processo que esteja na fase de julgamento, é possível a partir do *workflow* (Figura 18) a marcação do julgamento, a realização do julgamento e por fim o aguardar da sentença a ser proferida. Com a sentença proferida o processo fica a aguardar o trânsito em julgado, podendo ainda ser suscetível de recurso. Caso o prazo para interposição de recurso seja atingido e não houver a entrada de nenhum recurso, o processo tramita para transitado em julgado, ficando a aguardar o prazo que por sua vez despoleta a transição para arquivamento acabando o processo por terminar e consequentemente o *workflow*. Mas mesmo com um processo arquivado há sempre a possibilidade de ser interposto um recurso extraordinário. Ou seja, do ponto de vista processual não existe um estado terminal e último, findo o qual a instância é desativada e nunca mais invocada.

Para mais detalhes, o modelo visual completo pode ser consultado em Anexo A – Modelo conceptual do Wf. Proc. Penal.

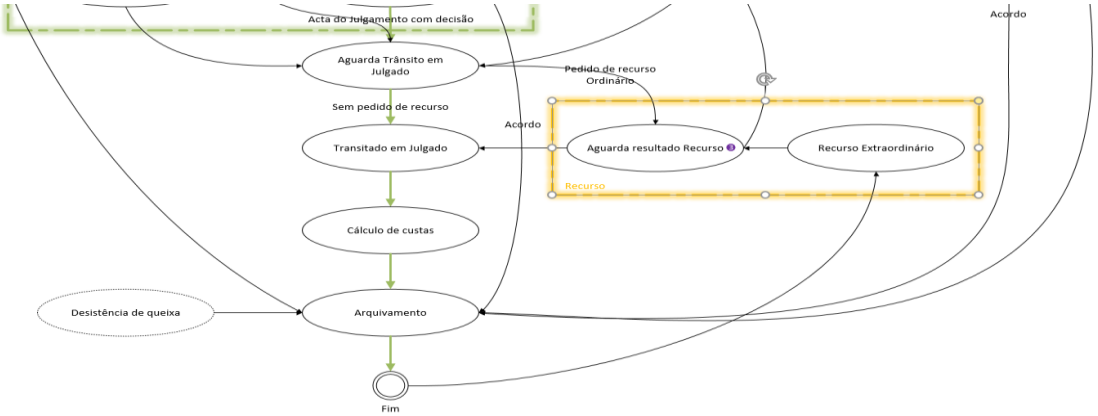


Figura 19: Modelo conceptual do workflow do processo penal, continuação, parte III.

3.2 Arquitetura do sistema em uso

Apresentar a arquitetura do sistema em uso, torna-se necessário para compreensão das alterações a ser feitas no caso prático de adaptação do novo sistema de *workflows* a ser abordado nos próximos capítulos deste estudo. Desta forma, e numa visão simplificada, pode-se dizer que o SIJ simula uma arquitetura MVC própria, assente em 3 camadas (Figura 20), onde a vista, projeto web MJCVWebApp, gere as requisições de forma dinâmica para a sua apresentação aos utilizadores. Como controlador nesta analogia a arquitetura MVC, onde passa todas as interações da vista com a camada Lógica, tem-se o projeto livraria de classes MJCVWebCode. Por outro lado, a camada Lógica disponibiliza uma lógica comum independentemente das aplicações que dela dependem, por ser desenvolvida de forma modular. Este abarca a subcamada de dados que assenta no sistema de gestão de base de dados (DBMS) Microsoft SQL Server.

Passando para uma descrição detalhada da arquitetura, constata-se a existência de uma outra subcamada na constituição da camada Lógica, neste caso os serviços de *workflows*. Os serviços de *workflows* têm como objetivo principal a gestão de procedimentos envolvidos nos processos judiciais. Desta forma, é da responsabilidade desta camada gerir todas a requisições referentes aos estados, transações e interações que podem ocorrer em cada processo judicial. É igualmente da responsabilidade deste, a definição de papéis de utilizadores que possam interagir com um processo numa determinada fase bem como as várias decisões possíveis mediante a fase processual.

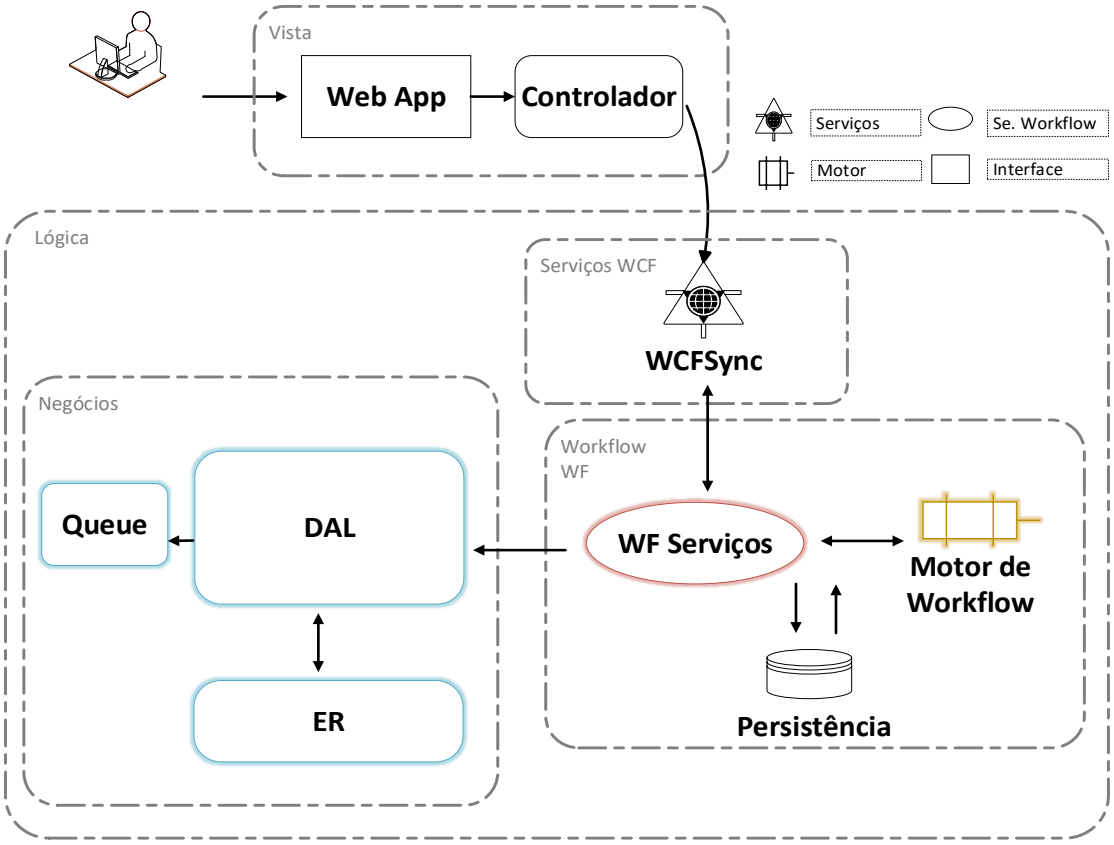


Figura 20: Arquitetura conceptual do sistema em uso, relativo a comunicação dos dados e workflow.

A camada Lógica (Figura 20) é a mais complexa, devido a coexistência de outras subcamadas (Negócios, Serviços WCF e Workflow WF) que compõem o sistema, tendo cada uma responsabilidades e funções destintas. A camada Vista interage com os serviços de *workflows* a partir do serviço de comunicações designado no sistema de WCFSync (da camada Serviços WCF), desenvolvido exclusivamente para gerir as comunicações entre *workflows* e também com a subcamada (onde se encontram a camada de acesso aos dados (DAL), o Modelo Entidade-Relação (ER) e a Queue) que gere a lógica de negócios. Este é responsável por direcionar os pedidos provenientes da aplicação web ou de outras instâncias para os serviços de *workflows* que por sua vez os enviam para o motor.

A lógica da subcamada Workflow WF, está dividida em três partes, sendo a Persistência onde estão os metadados dos *workflows*, os serviços de *workflows* WF Serviços onde constam os *workflows* modelados e prontos a ser executados e por fim o motor de *workflows*.

O motor é o responsável por gerir as instâncias dos *workflows* criados. Recebe os pedidos provenientes do serviço de comunicação, e mediante o tipo de pedido poderá instanciar um novo *workflow* ou readicionar a informação contida no pedido ou ainda pode fazer o resumir dessas mesmas instâncias. Por fim, ainda referente a interação entre os vários componentes da subcamada Workflow WF, tem-se a Persistência responsável por persistir as instâncias criadas e os metadados, permitindo ter dois modos diferentes para gerir as instâncias em execução, podendo ser mantidas em memória ou serem guardadas, por exemplo, numa base de dados relacional.

Para finalizar, tem-se na camada Lógica, a subcamada Negócios, que é responsável por gerir todos os pedidos à persistência de dados, estando dividida entre DAL, o modelo (ER) e a Queue. A DAL é responsável pela tradução das estruturas de dados, bem como por todas as ações sobre a persistência enquanto o modelo ER é responsável pela representação abstrata da estrutura das bases de dados do SIJ utilizando a tecnologia EntityFramework (mapeador de objeto-relacional - ORM, que permite aos programadores trabalhar com base de dados usando objetos .NET [59], [60]).

Outro componente da subcamada Negócios é a Queue. Este é responsável por gerir a execução dos eventos no sistema. Esses eventos são tarefas assíncronas realizados pelos utilizadores, que ficam na fila de espera para serem executados mediante a regra FIFO.

3.2.1 Serviços auxiliares em uso

Para além do serviço de comunicação, o sistema em uso possui dois outros serviços utilitários que desempenham um papel importante no que toca a persistência dos *workflows* e na recuperação e atualização de informações referentes a cada *workflow*, tanto em tempo de execução bem como quando persistidos, sendo eles GlobalPersistenceUpdate e GlobalListerProps.

A GlobalPersistenceUpdate permite a criação da instância de *workflow* mediante o tipo, a conversão do *workflow* antigo, a mudança de estados, abortar o *workflow* em execução, força o libertar do *workflow* da memória para a persistência, bem como pode listar os diversos tipos de *workflow* que constam na base de dados de persistência.

Por outro lado, o serviço GlobalListerProps permite obter em tempo de execução, as possíveis explicações de despachos mediante o estado de cada instância e também lista os diferentes estados de um determinado *workflow*. Estes serviços permitem a existência de múltiplos motores de *workflow* (até um por tipo de *workflow*) e fazem a comunicação com estes, tornando o acesso aos motores dos tipos de *workflow* completamente transparentes para a aplicação cliente que os utiliza.

Os serviços utilitários encontram-se implementados na camada de Negócio (Figura 21) interagindo com as interfaces do sistema da camada Vista e o controlador, mediante as requisições feitas.

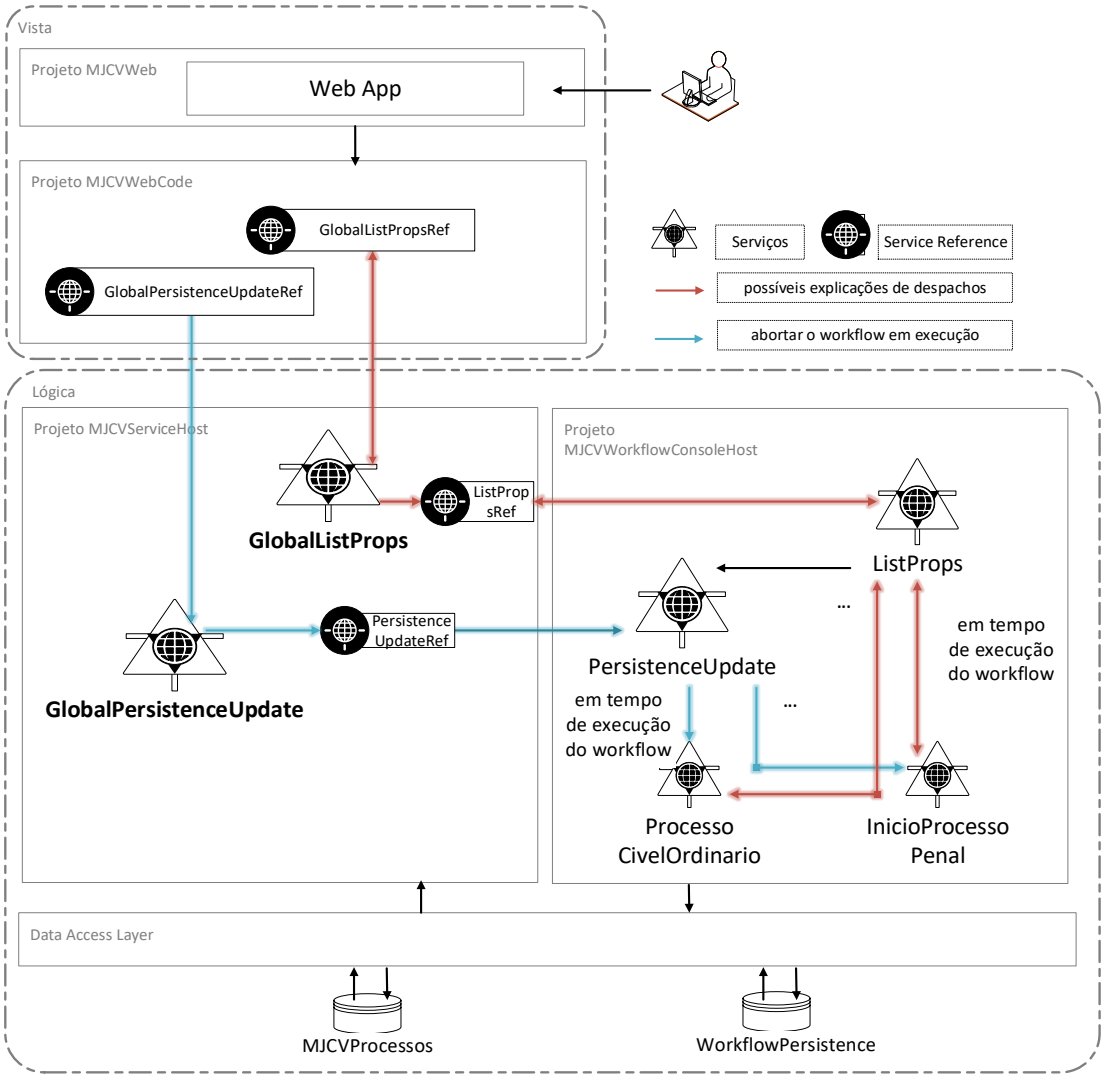


Figura 21: Arquitetura conceptual da interação dos serviços utilitários em tarefas como "possíveis explicações de despachos" ou "abortar o workflow em execução".

Tal como a apresentação dos serviços utilitários, antes de terminar esta secção, é importante a partir de uma visão conceptual, apresentar como é feita a interação dos diferentes serviços que compõem o sistema. Suponha-se a interposição de um documento de despacho ou requerimento no processo penal. Esta interação inicia-se com a redação de um despacho ou requerimento (Figura 22), a partir da interface da instância web. Este passa pela instância que faz de controlador, onde a partir deste é feita a referência ao serviço de *workflow* Documentos

em Versões utilizando o WSDL (formato XML para descrever os serviços como um conjunto de pontos finais [61] que operam em mensagens que contêm informações orientadas a documentos ou orientadas a procedimentos [62]).

Após o documento ter sido dado como terminado, o serviço de *workflow* Documentos em Versões faz referência ao serviço de comunicação que se encontra implementado no projeto de livraria de classes MJCVHosting e executado na instância de MJCVServiceHost. Este serviço de comunicação após receber o documento terminado, envia-o para o outro serviço de *workflow* Requerimento e Despacho (a interação número 2) que gere os despachos. Tendo o documento sido finalizado, o serviço de *workflow* Requerimento e Despacho faz referência ao serviço de comunicação para que este possa dar continuidade (a interação número 3). Mediante o tipo de documento redigido, o serviço de comunicação invoca o serviço de *workflow* do processo penal (interação número 4).

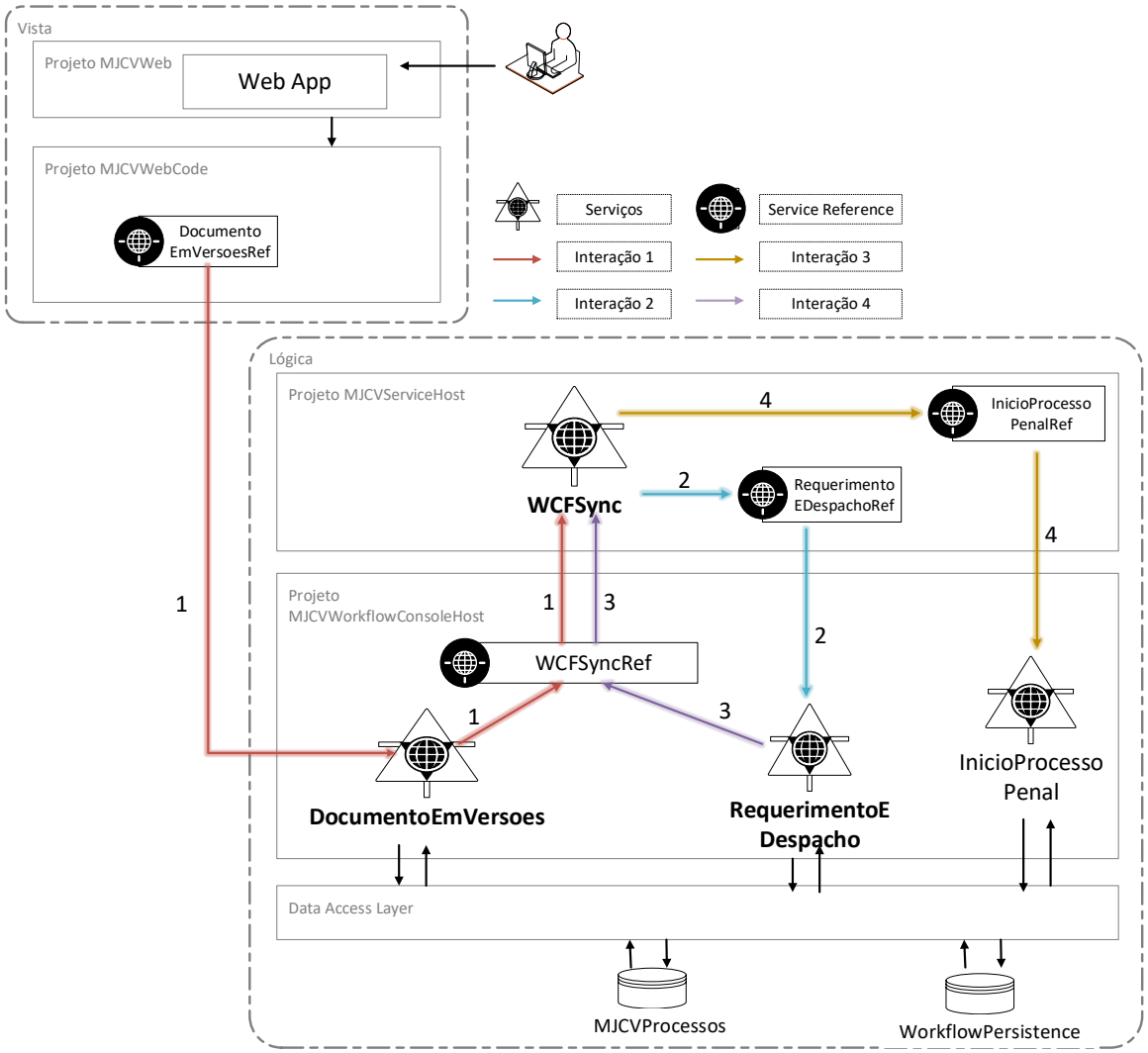


Figura 22: Arquitetura conceptual da interação dos diversos serviços a quando da entrada de um despacho ou requerimento no processo penal.

Com o documento no destinatário final, este é processado e anexado ao processo referente, de acordo com a informação da instância do processo e da instância do *workflow* que o processo está associado, despoletando-se eventualmente uma tramitação de fase ou estado no *workflow*.

3.3 Principais problemas advindos da abordagem em uso

No decurso do desenvolvimento e principalmente com a entrada em produção do SIJ, foram detetados alguns problemas que têm vindo as ser sentidos e motivados pela utilização desta tecnologia.

3.3.1 Desempenho e recursos computacionais necessários

O SIJ solicita a terceiros uma infraestrutura como serviço (IAAS) assente em máquinas virtuais. Esta infraestrutura operacional tem como base um *cluster* de base de dados composto por diversas máquinas, utilizando a DBMS Microsoft SQL Server. Faz parte também dessa infraestrutura, uma máquina onde são alojados os serviços de *workflows* e outros serviços auxiliares. Este possui ainda uma máquina onde está instalado o servidor de aplicação de serviços de informações da internet (IIS) que faz a gestão das requisições a instância web, na interação com os restantes componentes do sistema. A máquina onde encontra alojado os vários serviços, possui o sistema operativo da Microsoft Windows Server 2012 de 6 Gb de RAM.

Desta forma, ter uma infraestrutura como serviço possibilita uma escalabilidade tanto horizontal como vertical, mas este por ser disponibilizado por terceiros implica custos financeiros adicionais para o SIJ em caso de aumento da capacidade da infraestrutura. Neste sentido a solução e o que se recomenda é ter um sistema eficiente e que utilize os recursos de forma coerente, recorrendo apenas a quantidade de recursos necessários mediante a carga computacional que lhe é imposto.

Mas não é isto que tem acontecido muitas vezes com o sistema. Constantemente a equipa operacional tem-se deparado com a necessidade de tempos em tempos parar os serviços de *workflows* sobretudo o do processo penal, pois este tem alocado uma quantidade de recursos fora do comum, monopolizando em alguns momentos toda a máquina referente aos serviços (Figura 23).

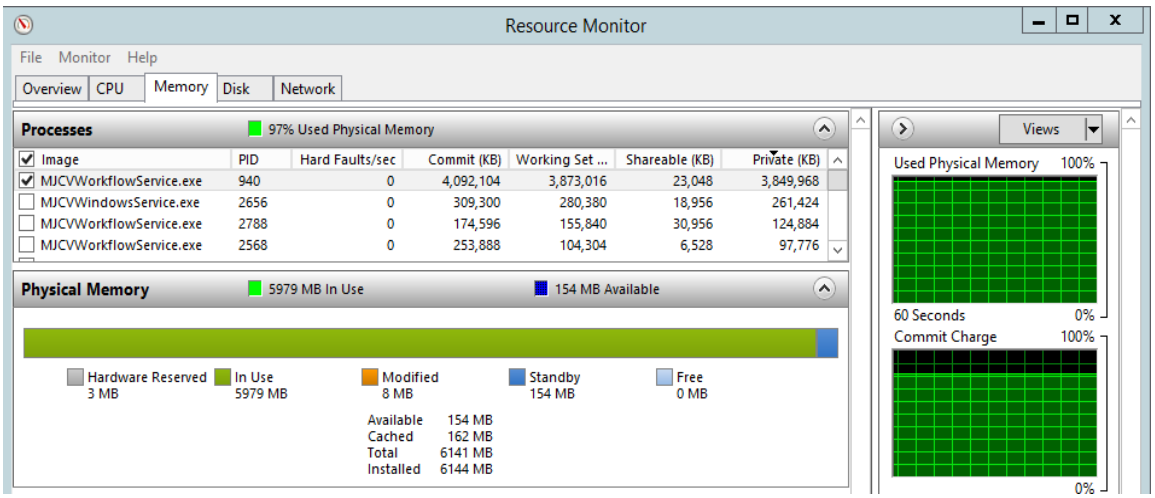


Figura 23: Recurso (memória) utilizado pelo serviço de workflow do processo penal.

O mesmo acontece com o CPU que quando este serviço aloca quase a total capacidade de memória, também este fica a trabalhar no seu limite máximo, como pode se constatar na figura (Figura 24).

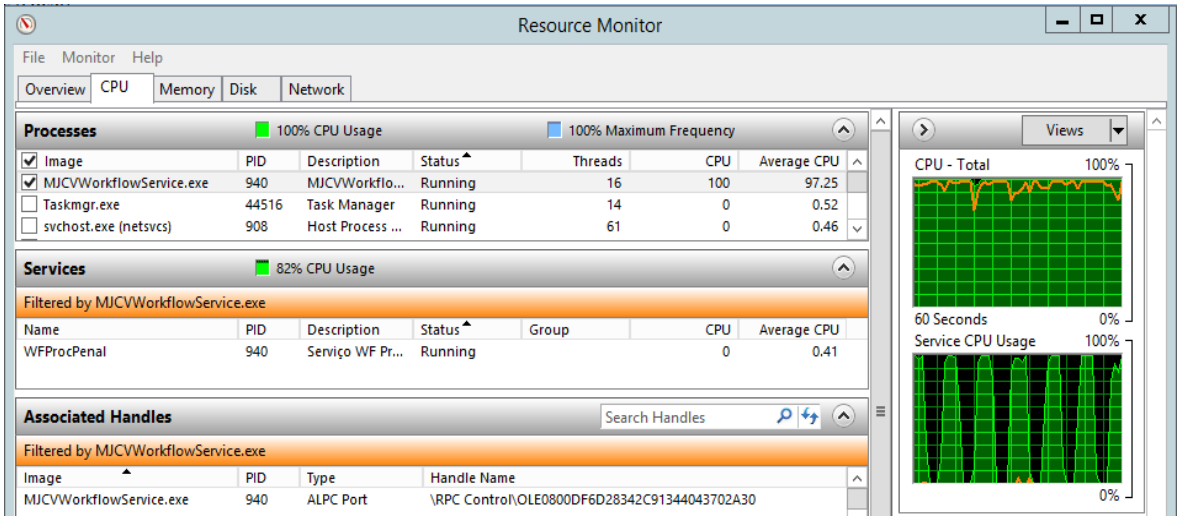


Figura 24: Recurso (cpu) utilizado pelo serviço de workflow do processo penal.

3.3.2 A evolução do WF

Advindo da evolução do WF, foi perdida a capacidade de (re)criar um determinado processo num qualquer ponto do *workflow* (Figura 25), bem como a possibilidade de administrativamente alterar o ponto de execução atual do processo para um ponto anterior ou posterior (Figura 26). Nesta figura é possível constatar a propriedade do WF 3.5, que permite criar um determinado *workflow*, independentemente do ponto que o *workflow* se encontre, acabando este por deixar de ser possível com a evolução.

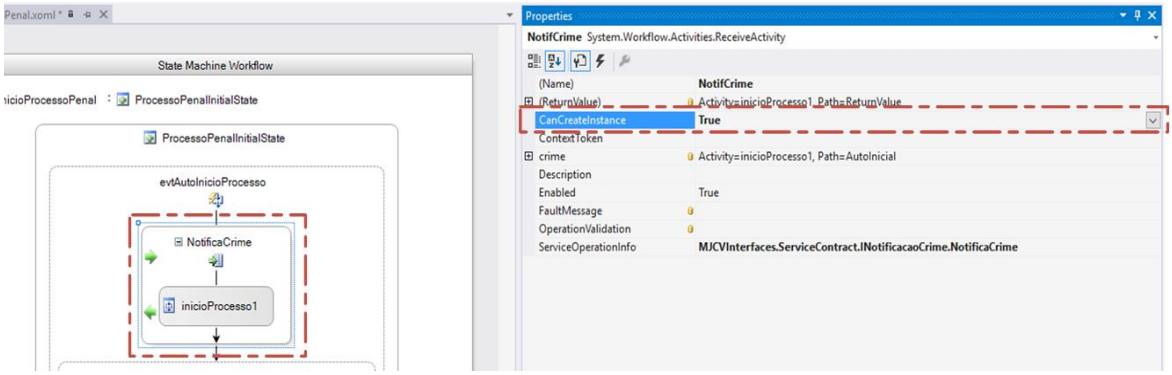


Figura 25: A propriedade que permite definir a criação de uma instância em qualquer ponto do workflow.

Outro ponto que deixou de ser possível com a evolução do WF, é a possibilidade de administrativamente alterar o ponto de execução. Este é feito (Figura 26) utilizando uma das interfaces dos serviços auxiliares apresentado na secção 3.2.1, que emparelhados nas atividades pré-concebidas de WF (*receive*) e utilizando uma das interfaces deste serviço, acede ao *workflow* em execução e efetua a alteração num determinado ponto.

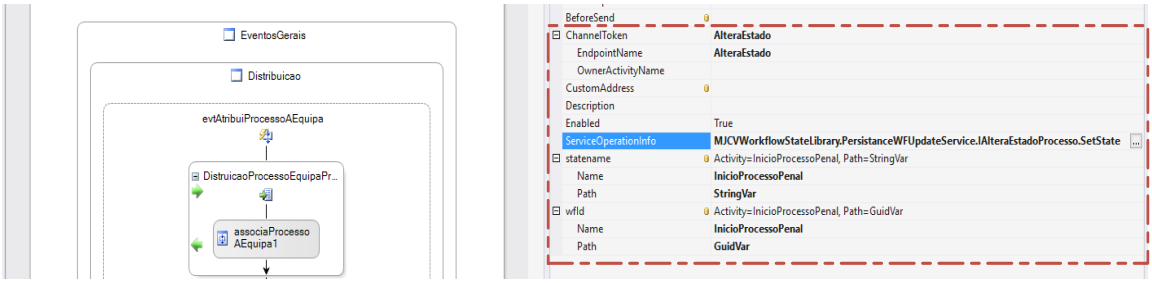


Figura 26: Exemplo da utilização de interfaces de serviço personalizados que permitem alterar o ponto de execução atual do processo para um ponto anterior ou posterior.

Sabendo que estas necessidades não se coadunam com os motores tradicionais, ainda que existam na WF 3.5 ainda em utilização e dada a necessidade de atualização do motor (para resolver algumas das questões identificadas) bem como a supressão das funcionalidades aqui referidas, tornou-se urgente a procura de novas soluções.

3.4 Interface de utilizador

A interface de utilizador do SIJ foi alvo de estudos e discussão entre a equipa de desenvolvimento e a comissão de acompanhamento, estando dividida em diversas áreas funcionais. Este é constituído pela Secretaria Judicial, Secretaria do Ministério Público, Portfólio, Agenda, Mensagens, Estatística, Pesquisa e a Documentação.

Para aceder ao sistema é preciso fazer autenticação mediante o cargo do utilizador. Como pode-se constatar (área A, Figura 27), este solicita a inserção do utilizador e a palavra-chave para fazer a validação. Para além dos campos para inserir as credenciais de acesso, a página disponibiliza um menu (área B, Figura 27) com informações referentes aos termos de utilização e as políticas de privacidade do SIJ.

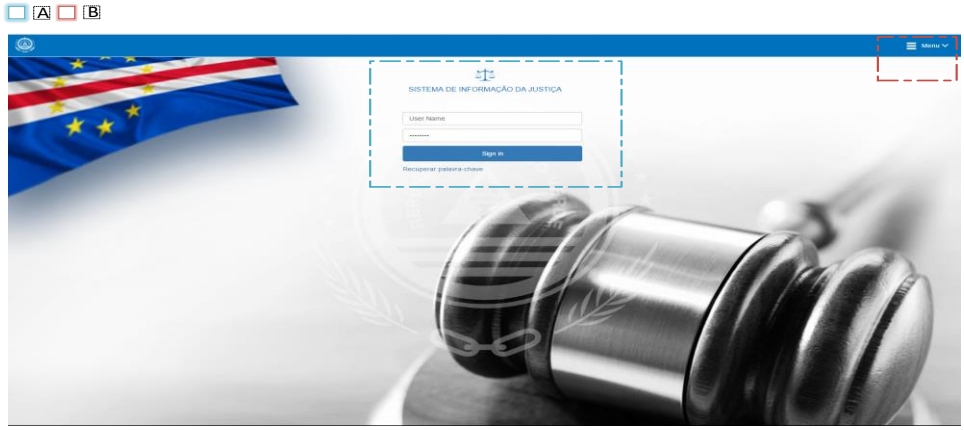


Figura 27: Página de autenticação.

Esta interface de utilizador adapta-se mediante o cargo que o utilizador desempenha no sistema judicial. Desta forma, após a autenticação o conteúdo a apresentar vai deferir de acordo com o cargo. Com a autenticação bem-sucedida, é possível navegar no sistema, sendo uma das funcionalidades assente na divisão das secretarias entre Secretaria Judicial (Figura 28) e a Secretaria do Ministério Público (Figura 29).

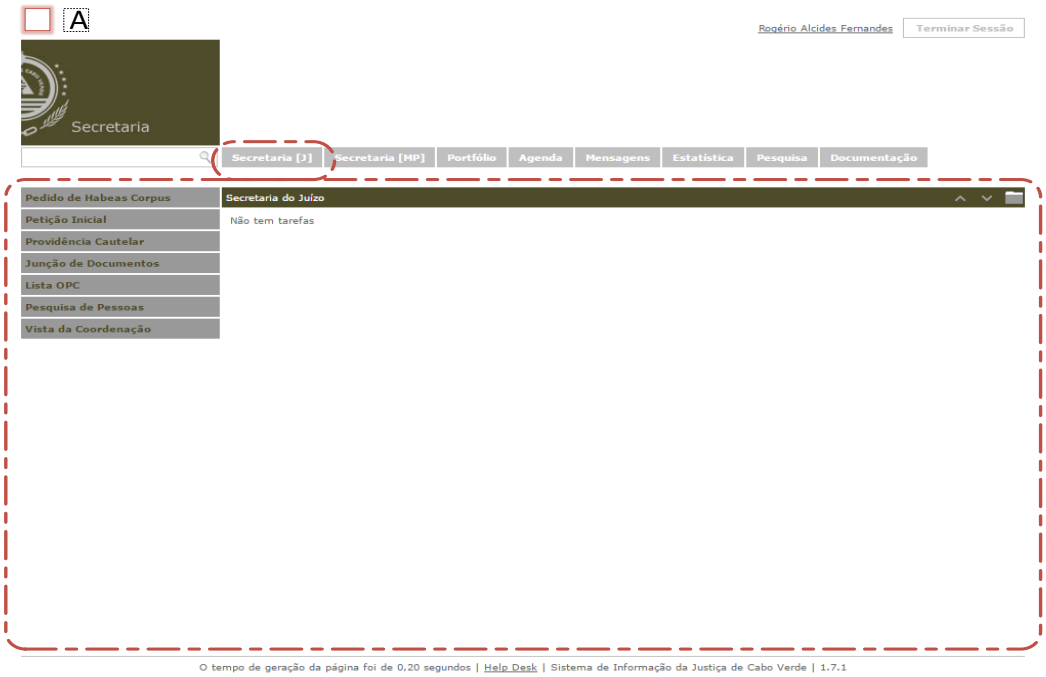


Figura 28: Área da Secretaria Judicial.

Nas secretarias, é possível realizar ações que são efetuadas nas secretarias físicas dos tribunais. Podem ser realizadas ações que se relacionam diretamente com a tramitação processual, tais como a inserção de um Auto, pedido de *Habeas Corpus*, bem como entrega de documentos para serem anexados a um processo. É igualmente possível executar funcionalidades que não se relacionam diretamente com a tramitação processual, tais como a gestão de ausências de magistrados ou a definição do pessoal de turno.

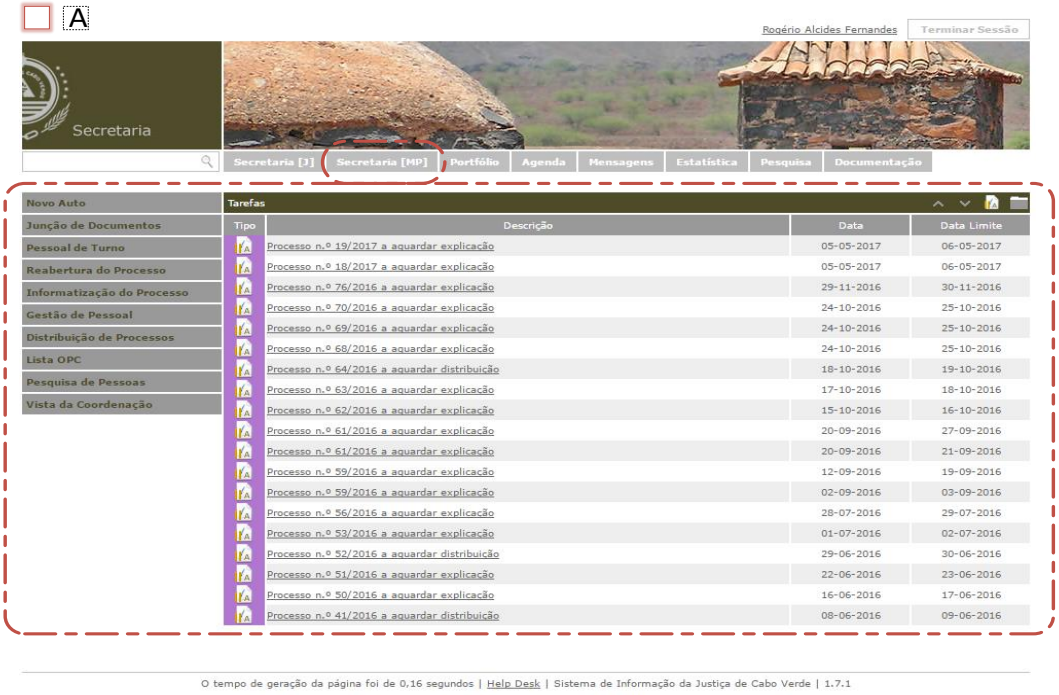


Figura 29: Área da Secretaria do Ministério Público.

Antes de abordar a constituição do portfólio é importante referir que na agenda é possível consultar as datas relacionadas com os processos ou até mesmo utiliza-lo como agenda privada. Nas mensagens, o utilizador poderá receber as notificações relativas às suas tarefas, bem como enviar mensagens privadas. Nas estatísticas é possível extrair dados estatísticos. A Pesquisa permite, pesquisa livre sobre os dados dos processos. Por fim a documentação apresenta a legislação, bem como a documentação relativa ao SIJ.

O portfólio é outra secção de importância vital para o sistema, onde grande parte das tarefas do processo são realizados (Figura 30). Este contém os processos aos quais o utilizador tem acesso, e poderá realizar ações, mediante o seu perfil e o estado do processo.

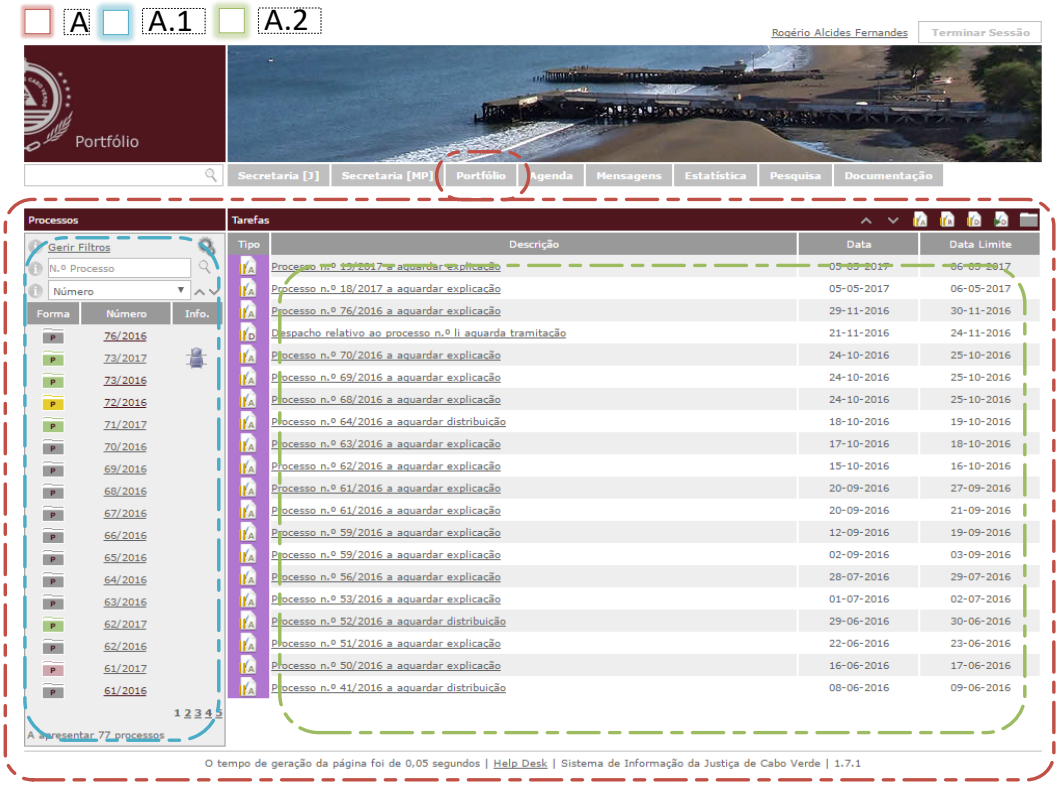


Figura 30: Página do portfólio.

O portfólio é a área central (Figura 30) da interação dos utilizadores com os processos sendo considerado como a área mais sensível do sistema. Desta forma, são várias as tentativas de aperfeiçoar e disponibilizar ao utilizador final uma interface amigável facilitando a realização das suas tarefas cotidianas no sistema.

Este encontra-se dividido em duas outras secções (Figura 30). Na área A.1 estão listados os processos a que o utilizador tem acesso, enquanto que na outra secção (área A.2 da Figura 30) tem-se a listagem das tarefas por realizar.

Cada tipo de processo contém informação codificada através de ícones e texto (área A.1 da Figura 30). Para representar uma forma de processo, mediante a sua atribuição, a cor varia, advindo das cores das capas dos volumes dos processos. Neste caso para processos sumário o ícone da pasta apresentado na secção à esquerda

aparece de cor amarela, para abreviados da cor vermelha, ordinário verde, transação lilás, e cinza para processos sem forma atribuída.

Outro aspeto importante na forma como os processos são representados, tem a ver com a numeração (área A.1 da Figura 30). Fez-se alterações na legislação para estandardizar a numeração processual. Assim, a numeração passou a ser sequencial e única a nível nacional, mediante o ano, com base na data de entrada do processo no sistema. No entanto, para processos digitalizados, o número do processo advém predefinido da secretaria do tribunal que fez a digitalização.

Ainda no *portfolio*, após a seleção de um processo, as tarefas são substituídas por informação relativa ao processo (as áreas A.1 e A.2 da Figura 31). Mediante o perfil e o estado que o processo se encontra é possível efetuar ações e analisar despachos ou requerimentos pendentes ou até mesmo fazer a sua explicação. Este é possível a partir da secção Ações do *portfolio* representado na área A.4 da (Figura 31).

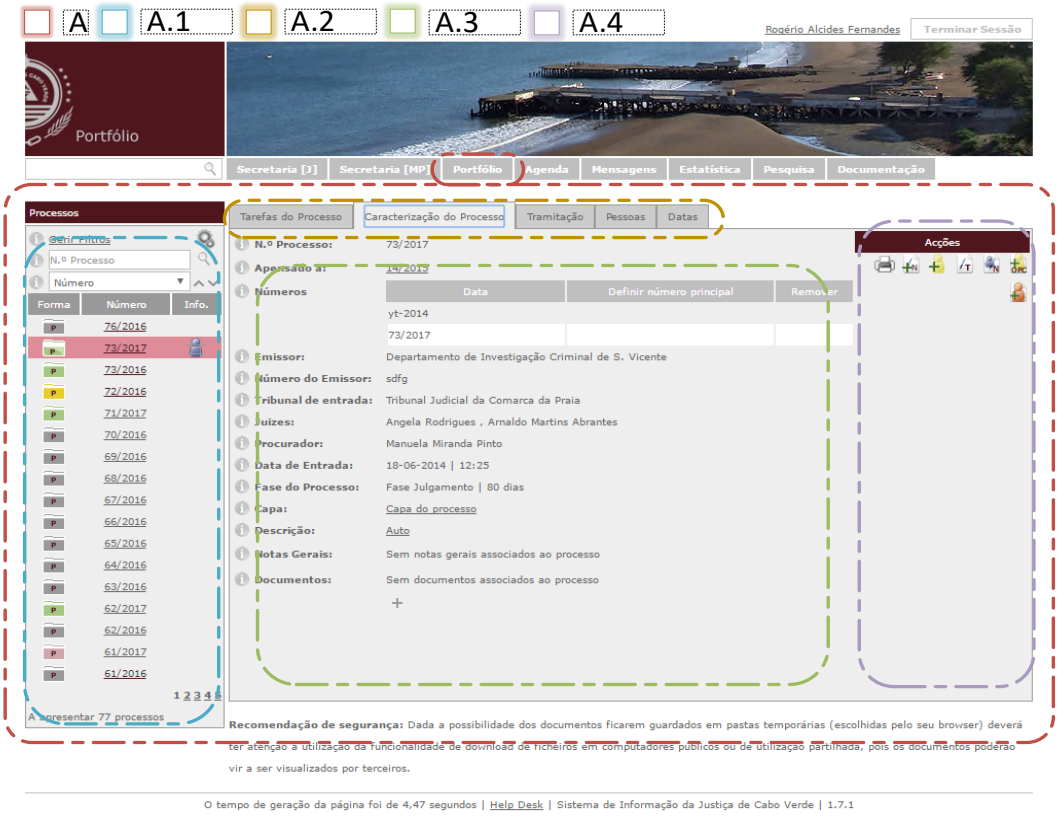


Figura 31: Página do portfolio depois de selecionar um processo específico.

3.5 Resumo

Neste capítulo apresentou-se o sistema atual em uso e problemas relacionados. Primeiramente abordou-se a tipologia dos *workflows*, destacando os principais *workflows* que são a base do caso prático. Incidiu-se na lógica associada a cada um desses *workflows* sendo o do processo penal moldado pelas várias formas que este pode ganhar. Essas formas incluem processos abreviados, sumário, ordinário e transação.

Seguidamente apresentou-se a arquitetura do sistema em uso, relativo à comunicação dos dados e *workflow*. Essa arquitetura é composta por camadas principais e suas subcamadas. Ainda na apresentação da arquitetura abordou-se a importância dos serviços auxiliares e a interação dos diversos serviços em eventos despoletados no sistema de *workflows*.

Conhecida a arquitetura e os seus serviços auxiliares, tornou-se pertinente apresentar os problemas sentidos no sistema. Desses, destacam-se os provocados pela evolução da tecnologia de base e o desempenho e recursos necessários para o seu funcionamento. Com a evolução desta tecnologia foi perdida a capacidade de (re)criar um determinado processo em qualquer ponto do *workflow*, bem como a possibilidade de administrativamente, alterar o ponto de execução atual do processo para um ponto anterior ou posterior.

Para finalizar o capítulo foi apresentada a interface do utilizador do SIJ, focando nas principais áreas do sistema ao qual esse tem uma relação direta com as ações que depende de eventos externos para a transição entre estados, normalmente associados a decisões humanas no sistema de *workflows*.

4. Redesenho/refatoração de códigos WF e a passagem para serviços granulares

Neste capítulo, são apresentadas as diretrizes definidas para a construção dos serviços granulares. A criação dos serviços consistiu em retirar a lógica de dentro dos *workflows* anteriores para serviços, permitindo o acesso a estes independentemente da plataforma, localização ou ambiente que os rodeia. Estes encontram-se definidos no novo projeto adicionado a solução, designado por MjcvServices dentro da camada Vista.

A retirada da lógica foi possível em parte com a refatoração e simplificação de códigos contidos nos *workflows*. A refatoração é o processo de alterar um sistema de *software* de tal forma que não altera o comportamento externo do código, melhorando a sua estrutura interna [63]. A simplificação permite melhorar a legibilidade do código e facilita na sua manutenção, onde este permitiu remover códigos não necessários (YAGNI- *you aren't gonna need it*, uma afirmação usada para referir a "supressão" de certas capacidades no software que podem vir a ser necessárias só no futuro [64]) e códigos que não são utilizados (códigos mortos).

A outra parte da retirada da lógica de dentro dos *workflows*, consistiu na interpretação do código declarativo constante na definição de *workflows*. Assim sendo, foi possível a reutilização do código e da lógica que constam nas atividades. As atividades são os principais blocos de construção do WF, sendo um componente discreto e reutilizável desenhado para cumprir um propósito definido (Figura 32).

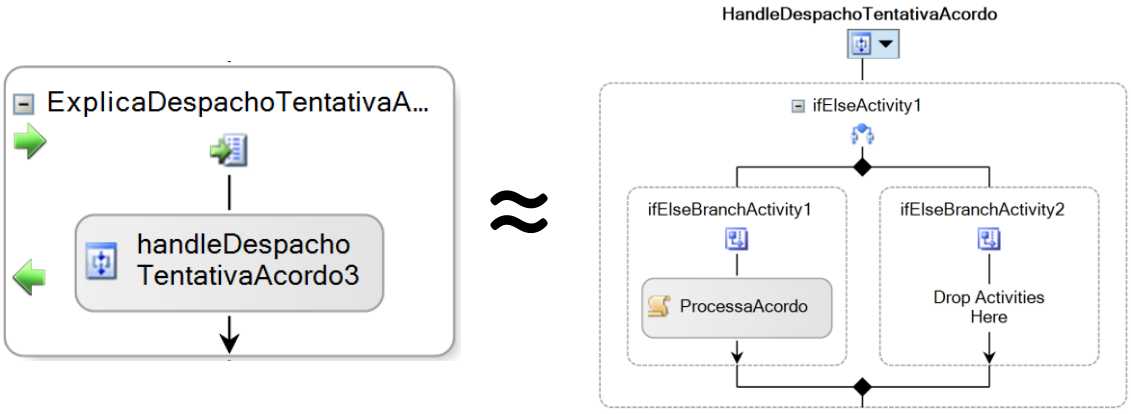
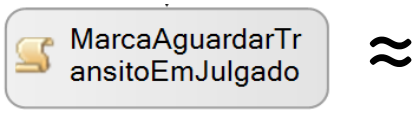


Figura 32: Exemplo de uma atividade e sua composição em WF 3.5.

As atividades podem conter atividades de códigos, que permitem a adição de código próprio ao *workflow*, sendo uma forma simples de criar atividades personalizadas. Estas atividades são colocados em um arquivo de "código ao lado" sendo compilados juntos com o *workflow* e executado de forma síncrona até a conclusão da execução [65] (Figura 33).

O WF inclui um conjunto de atividades pré-concebidos na sua estrutura, mas igualmente permite desenvolver atividades personalizadas para resolver problemas específicos [1], [66].



≈

```
1. private void MarcaAguardarTransitoEmJulgado(object sender,
   EventArgs e)
2. {
3.       MJCVInterfaces.BusinessObjects.Processo.SetFaseProcesso(
         "Aguardar Trânsito Em Julgado", DateTime.Now,ProcessID);
4. }
```

Figura 33: Exemplo de uma atividade de código personalizado em WF 3.5.

Abordando a arquitetura, apesar destes serviços granulares estarem definidos na camada Vista, a implementação encontra-se na subcamada Negócios na DAL. A interação entre as duas é suportada utilizando referências às bibliotecas de vínculo dinâmico (DLLs) do projeto onde consta a lógica implementada (Figura 34). Uma DLL é uma biblioteca que contém códigos e dados que podem ser usados por mais de um programa ao mesmo tempo. Ao usar uma DLL, um programa pode ser modularizado em componentes separados [67]. Esta separação garante o acesso aos modelos de entidade-relação mapeados na camada Lógica e às demais classes de objetos auxiliares definidos nesta camada, facilitando na fase de desenvolvimento do sistema e permitindo a retirada da lógica de dentro dos *workflows* e a reutilização dos componentes.

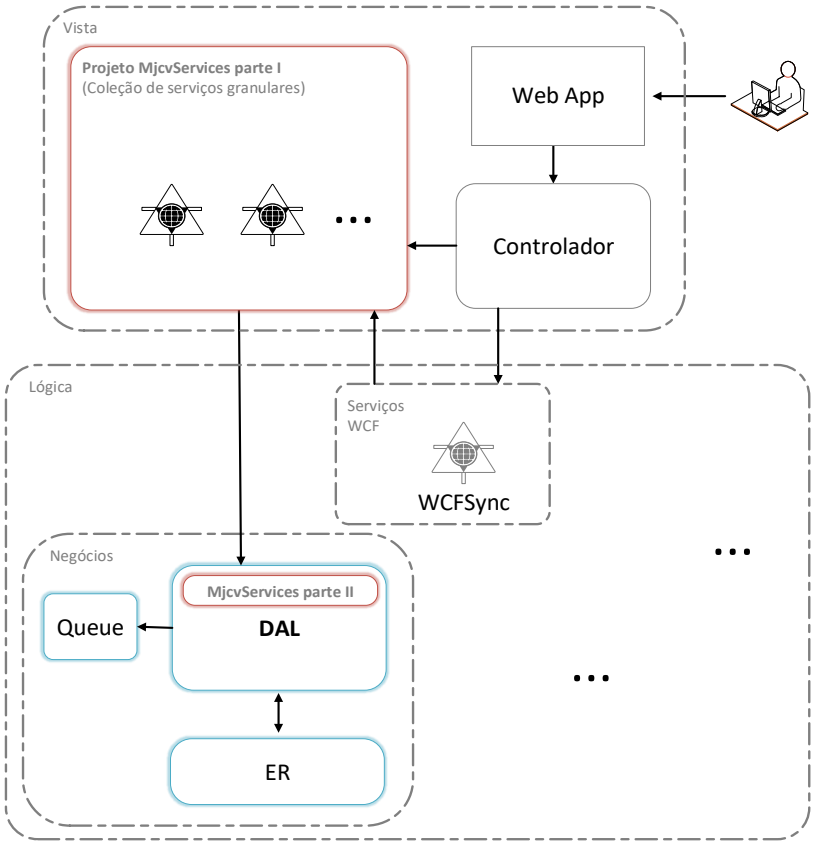


Figura 34: Arquitetura conceptual da integração do novo componente (serviços granulares) na solução do SIJ.

Com a retirada da lógica de dentro dos *workflows*, constituíram-se doze serviços distintos separados mediante a lógica associada. Cada serviço tem as suas próprias interfaces, sendo que alguns implementam mais do que uma interface de serviço. Estas interfaces e operações foram criados com base nas que eram utilizadas pelos

serviços de *workflows* anteriores. As interfaces descrevem as operações implementadas por ponto final do serviço exposto ao exterior [68]. Este define os formatos das mensagens e descreve como eles são trocados, além de mapear os métodos da classe para serviços WSDL, tipos de portas e operações. Por outro lado, as operações dentro das interfaces descrevem os procedimentos do serviço, que são os métodos que implementam as funções do serviço [69].

Nas secções seguintes são apresentados em detalhe esses serviços granulares, as suas interfaces, operações e sobretudo a lógica das atividades que lhes deram origem.

4.1 Atribuição de processos

O serviço de atribuição de processos permite fazer a distribuição do processo a equipas de procuradoria e juízo, bem como, a sua redistribuição. Este implementa a interface IAtribuicaoProcesso composto pelas operações de distribuição e de redistribuição de processos. Estas operações constam da programação declarativa contendo atividades condicionais e atividades personalizadas de WF (Figura 35 e Figura 36) que implementam a lógica em cada uma delas.

A lógica associada a distribuição da equipa de procuradoria a um processo, consiste em verificar em caso de processos urgentes, os procuradores que encontram de turno, sendo de seguida calculado o peso do processo para a sua atribuição (Figura 35).

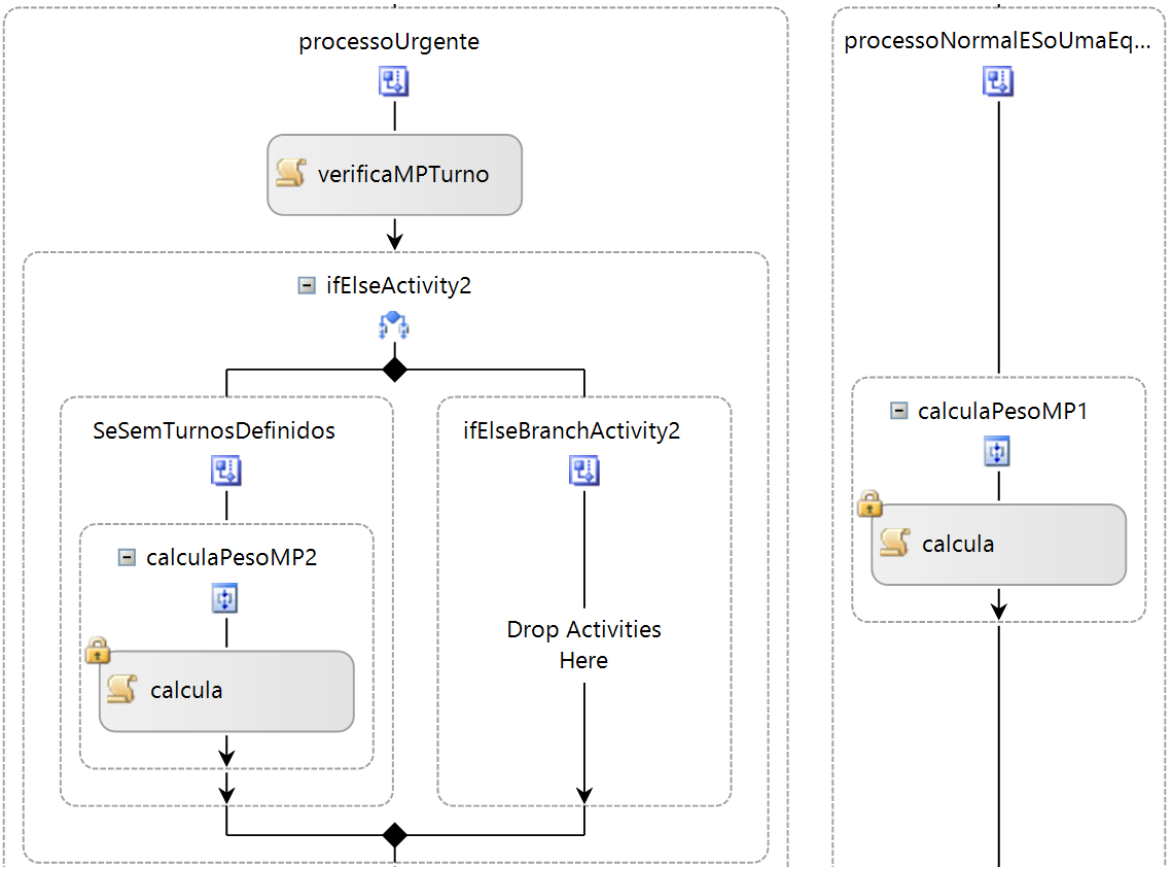


Figura 35: A programação declarativa implementada na atividade de associação do processo automaticamente ao MP.

Igualmente para as operações de associação da equipa do juízo, tanto para processos urgentes bem como para processos normais que não exigem a associação de juízo de turno, calcula-se o peso do processo (Figura 36) antes da associação de uma equipa de juízo.

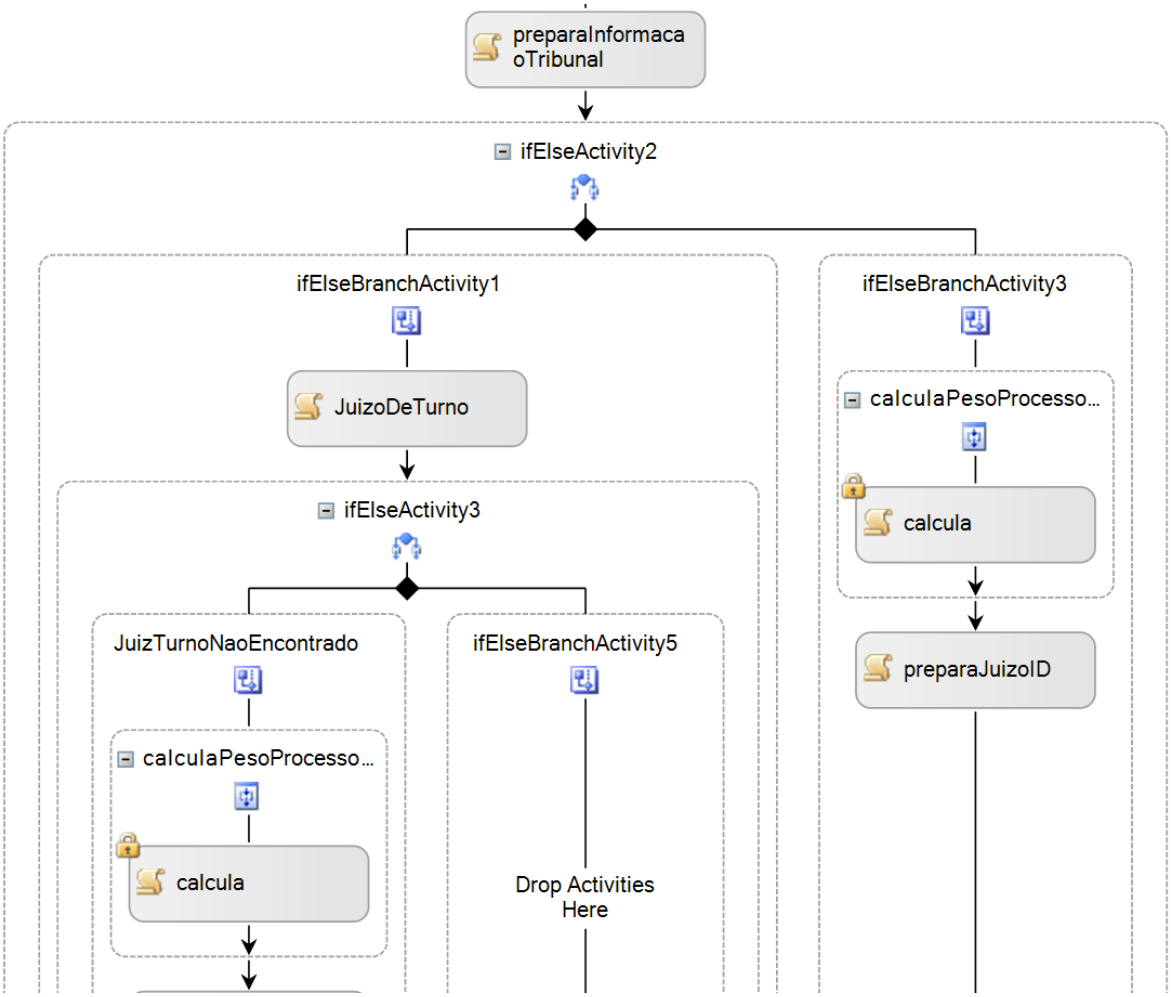


Figura 36: A programação declarativa implementada na atividade de associação do processo ao Juízo.

Toda a lógica constante nas Figura 35 e Figura 36, foi reutilizada neste serviço granular, tendo este sido atribuído o nome AtribuicaoProcesso.svc. A implementação encontra-se definida na subcamada Negócios.

4.2 Audiência de processos

O serviço granular Audiencia.svc, disponibiliza uma interface para a persistência da ata redigida na fase de julgamento, mediante a instância do *workflow*. Este implementa a interface IAudiencia contendo a operação de gravar a ata.

Uma ata redigida na fase de julgamento contem todas as principais ocorrências e protestos feitos tanto pela acusação bem como pelos defensores. Este igualmente pode conter uma sentença ou não. Tal como os outros serviços, este também teve como base a reutilização das atividades e atividades de código anteriormente

constantes da lógica dos *workflows* WF. O código e a lógica reutilizada encontram implementados na subcamada Negócios.

4.3 Auto de processo

Este serviço granular, assenta em receber o documento preenchido de auto de denúncia ou de flagrante delito que será persistido no sistema. Implementa a interface IAutPreenchido contendo a operação que recebe a ordem e o auto preenchido.

O processo de reutilização associado a este serviço, permitiu retirar e rearranjar o código anterior das atividades. As atividades condicionais foram substituídas por código procedimental e foram igualmente removidos atributos e propriedades que não são mais necessários.

Os vários serviços granulares, contêm dependente da sua lógica, uma instrução importante, que tem como suporte a definição constante no modelo visual. Este é a transição de um estado para o outro. Como referenciado na secção 3.1.3, a maior parte das transições são despoletadas por entradas de despachos e a sua explicação. Desta forma, pode-se constatar o exemplo de uma a instrução (Código 1, linha 4) que faz a chamada ao API do motor de *workflows*, associando os parâmetros mediante os definidos nos metadados do modelo.

```
1. using (WorkflowEngineComposerService.WorkflowEngineComposerServiceClient wfeng = new
   WorkflowEngineComposerService.WorkflowEngineComposerServiceClient())

2. {

3.    string dest =
   WorkflowEngineComposerClass.DevolveStateNameToTransitbyIDTransitionValue(wfen
          g.Transitions(ConverterGuidString.ToIntString(WorkflowInstanceId)),
          "","ExplicaDespachoAdiamentoJulgamento");

4.
   wfeng.ChangeState(ConverterGuidString.ToIntString(WorkflowInstanceId),destination);

5. }
```

Código 1: Trecho de código da mudança de estado.

4.4 Decisão do recurso de processo

O serviço granular de decisão de recurso, disponibiliza uma interface para a entrada do despacho de recurso final. Este implementa a interface IDecisaoRecursoProcesso contendo uma única operação.

A lógica do serviço (Figura 37), assenta em verificar se o documento é aceite. Caso contrário, é dado como indeferido e se não existirem outros recursos ativos é feito a tramitação do *workflow* para outra fase, estando o serviço implementado na subcamada Negócios.

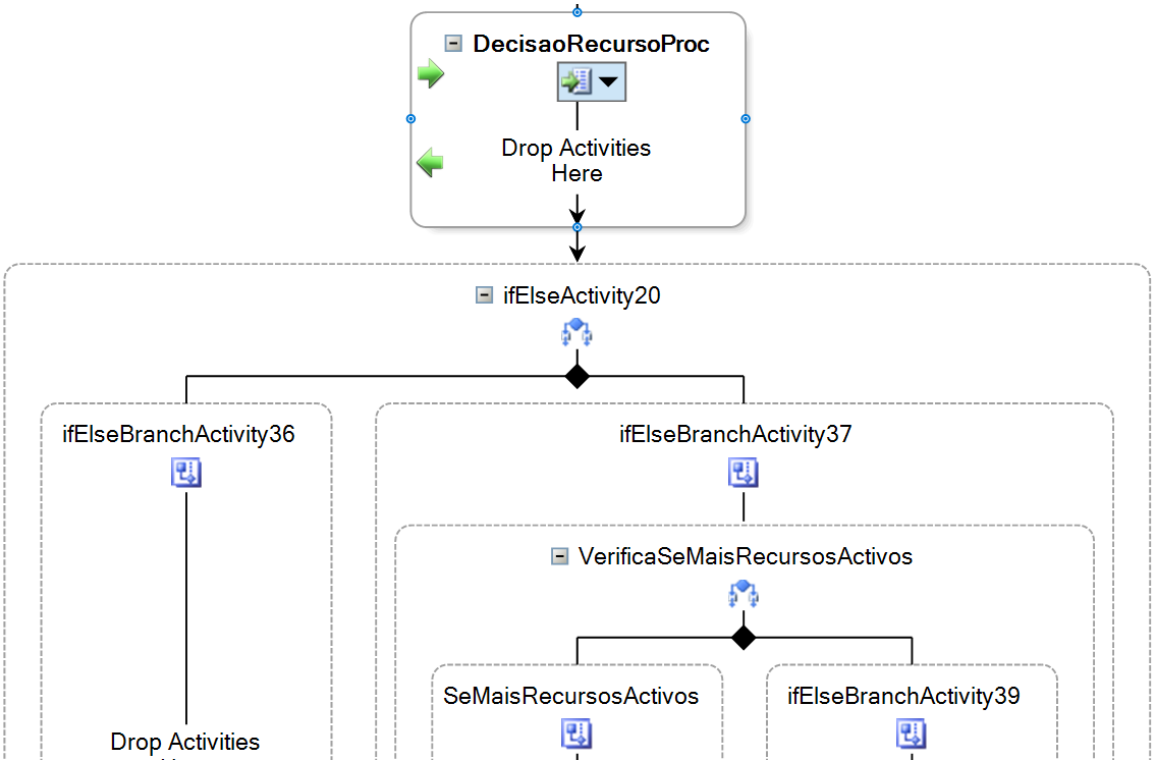


Figura 37: A programação declarativa implementada na atividade de decisão de recurso.

4.5 Devolução de auto de processo

O serviço granular de devolução de auto do processo, permite recuperar o auto ainda não terminado, retornando o documento para ser explicado. A explicação de um auto é o processo de associar os arguidos, crimes, ofendidos, testemunhas e outras informações referentes a um processo.

Este serviço implementa a interface IDevolveAuto contendo uma única operação, que recupera o auto mediante a instância do *workflow* associado ao processo, verificando a existência de documentos em produção para confirmar que este realmente se encontra por explicar.

Na tecnologia WF, as atividades são compostas por outras instruções e atividades que determinam a lógica associada a este e que executam as instruções.

Nessa nova abordagem assente em serviços granulares, o código contido na atividade de código, encontra-se implementado na subcamada Negócios, em pequenas partes de códigos procedimental.

4.6 Interposição de recurso de processo

Este serviço disponibiliza uma interface para entrada de recursos judiciais tanto ordinários como extraordinário. Implementa a interface IInterposicaoRecursoProcesso, que possui duas operações distintas, uma que trata da lógica de recurso ordinário e uma outra para recursos extraordinários. O ordinário pode acontecer quando o processo não tenha transitado em julgado ao contrário do extraordinário (secção 3.1.3).

Na programação declarativa (Figura 38), usa-se uma das atividades pré-concebidas (*receive*), para a disponibilização de uma interface que aceita o objeto específico.

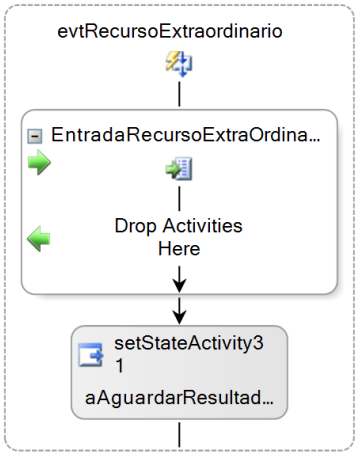


Figura 38: A programação declarativa do evento de entrada do recurso extraordinário composto pelas atividades pré-concebidas no WF (receive e setState).

A lógica deste serviço, assenta em executar o comando para a transição de estado do *workflow* na API do motor de *workflows*. Já a tarefa de persistir o documento é delegada ao *workflow* de documento em versões, que tendo o documento persistido, invoca um dos pontos finais deste serviço, a partir do serviço de comunicação.

4.7 Notificação de crime

O serviço granular recebe o auto preenchido, podendo ser de denúncia ou de flagrante delito. Este despoleta o início do processo penal, em processos inseridos no sistema a partir de um auto. O serviço implementa a interface INotificacaoCrime contendo a operação para autos de flagrante de delito e para autos de denúncia.

A lógica implementada herda a programação declarativa do WF (Figura 39). Este permite instanciar um *workflow* de processo penal invocando o ponto final da API do motor de *workflows* que retorna um identificador único ou a instância do *workflow* criado, que é de seguida associado a uma instância do processo.

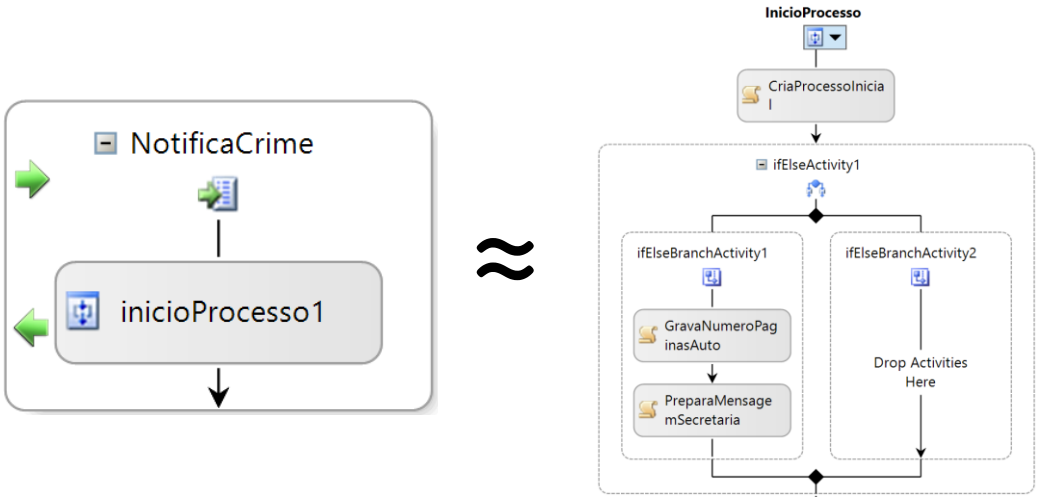


Figura 39: A esquerda a utilização da atividade personalizada de início de processo com a sua lógica definida a direita.

Ambas as operações recebem como parâmetro um objeto Auto, que possui as propriedades inerentes a este tipo de documento. A implementação completa deste serviço encontra-se na subcamada Negócios, em pequenas partes de códigos a par dos métodos auxiliares.

4.8 Reabertura de processo

O serviço reabertura de processo disponibiliza uma interface para a informatização de processos e uma outra para reabrir processos arquivados. Este serviço implementa a interface IReaberturaProcesso, contendo a operação que trata a lógica de informatização e também a operação para tarefas de reabrir processos.

A operação de reabertura de processo na programação declarativa, contém atividades personalizadas e atividades de código. Estas serviram para a implementação deste serviço. Por outro lado, a lógica da operação de informatização consiste no recebimento do objeto de informatização com os dados necessários para proceder à tarefa (criação de novo processo). Recebido o objeto é instanciado o *workflow* de informatização, sendo de seguida verificado o tipo de informatização, despoletando um novo tipo de *workflow* advindo do documento do processo informatizado (Inicio Processo Penal, Recurso etc…).

4.9 Separação de processos

Este serviço permite separar processos mediante a instância do *workflow* do processo em causa, implementando a interface ISeparaProcesso contendo uma única operação que trata a separação de processos. O código constante da atividade de código (a direita da Figura 40), foi reutilizada neste serviço, estando implementado na subcamada Negócios.

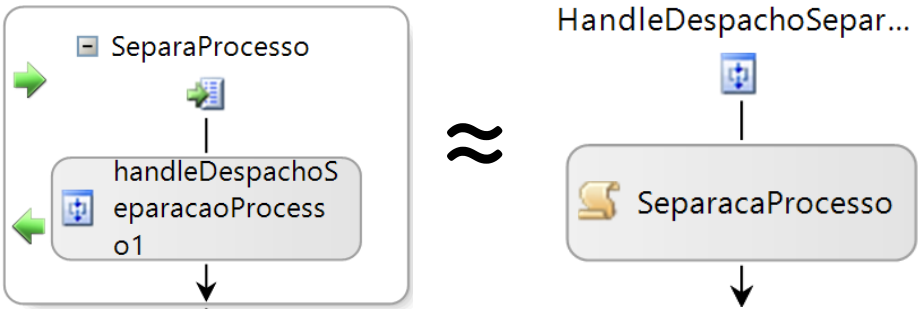


Figura 40: A esquerda a utilização da atividade personalizada de separação de processo com a sua lógica definida a direita.

4.10 Explicação de despacho

No serviço granular DespachoRecebido.svc, encontra-se implementada a lógica associada às explicações de despachos, sendo este a base para a tramitação do *workflow* do processo penal.

Na Figura 41, é apresentada a programação declarativa das operações de Despacho de Aceitação de Interrogatório e Despacho de Alteração da Acusação, sendo estes apenas alguns dos exemplos das operações apresentadas na Tabela 9. Além dos condicionais, esta programação declarativa também contém atividades de

código. Ambos foram utilizados na nova abordagem de programar as operações em serviços granulares fazendo a reengenharia da lógica e do código contido neles.

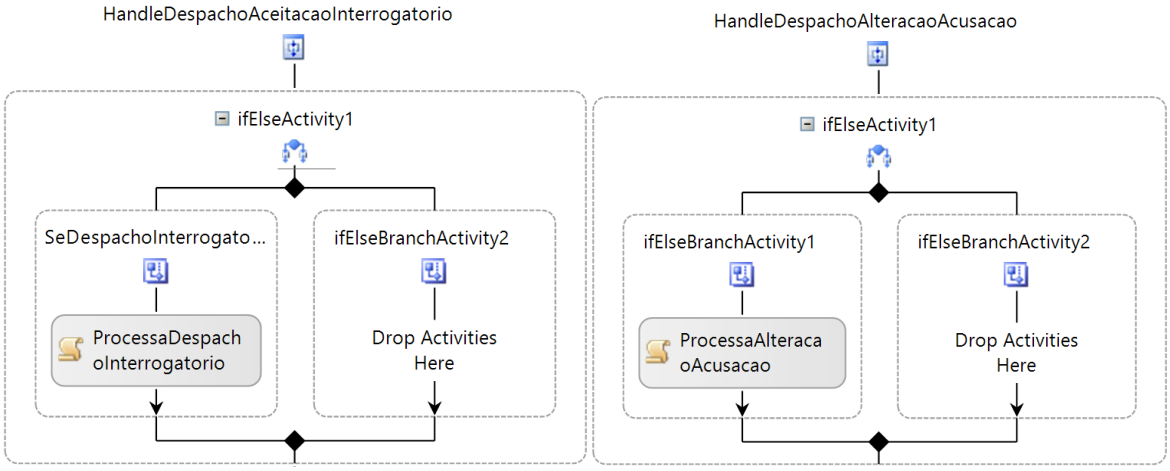


Figura 41: A programação declarativa do despacho de aceitação de interrogatório e alteração da acusação.

Este serviço implementa a interface IDecisao com as seguintes operações:

Tabela 9: Lista das operações implementadas pelo serviço DespachoRecebido.svc.

Operações	
ApensaProcesso	ExplicaDespachoArquivamento
ExplicaDespachoAberturaACP	ExplicaDespachoEsclarecimento
ExplicaDespachoACPNaoPronuncia	ExplicaDespachoPedidoAdvogadoOficioso
ExplicaDespachoACPPronuncia	ExplicaDespachoDecisao
ExplicaDespachoAdiamentoJTransaccao	ExplicaDespachoEntregaDocumentos
ExplicaDespachoAdiamentoJulgamento	ExplicaDespachoContestacao
ExplicaDespachoDecisaoJSRExtraordinario	ExplicaDespachoAlteracaoPenal
ExplicaDespachoDecisaoJSROrdinario	ExplicaDespachoAlteracaoRol
ExplicaDespachoMarcacaoJulgamento	ExplicaDespachoAlteracaoAcusacao
ExplicaDespachoPronuncia	ExplicaDespachoConstituicaoAdvogado
ExplicaDespachoRequerimentoTransac	ExplicaDespachoDecisaoJOuSentenca
ExplicaDespachoTentativaAcordo	ExplicaDespachoTentativaAcordoInstrucao
SeparaProcesso	ExplicaDespachoConstituicaoAssistente

4.11 Documento em versões

O serviço granular de documentos em versões, possui todas as operações anteriores do *workflow* do Documento em Versões. Permite guardar o documento interposto e devolver o documento ainda não terminado, ou seja, em produção. Permite igualmente alterar a versão do documento em uso e permite eliminar um documento que pode ter sido inserido por lapso. Cada uma das operações, contém uma lógica associada, estando esta implementada na subcamada Negócios. Tal como os serviços apresentados nas secções anteriores, as operações (Tabela 10) deste serviço foram inicialmente programadas de forma declarativa, onde por exemplo, a lógica da operação de guardar documento (Figura 42), estabelece as condições para se chegar ao objetivo pretendido.

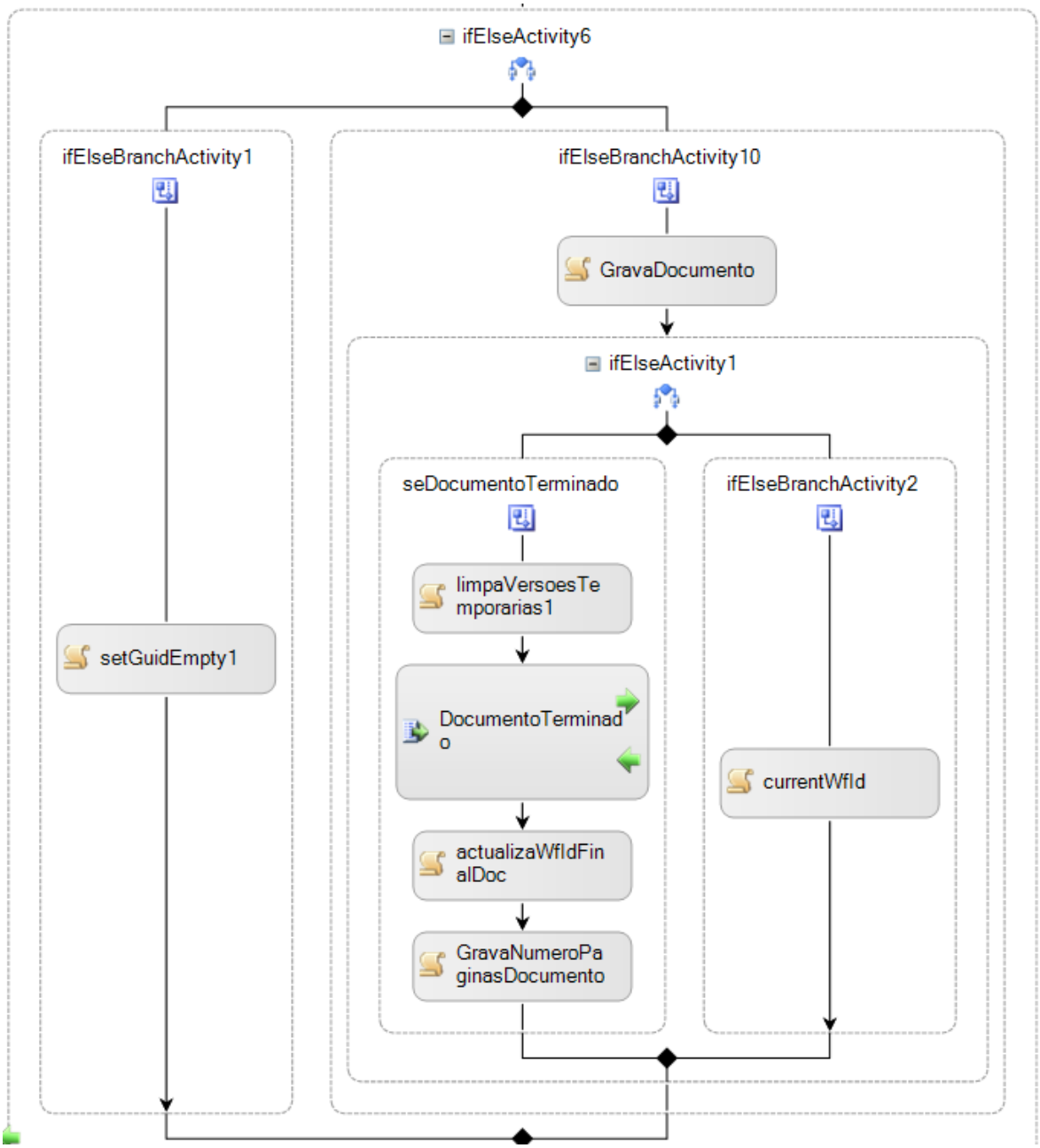


Figura 42: A programação declarativa do evento de guardar documento entregue, composto pelas atividades pré-concebidas pelo WF (send e ifelsebranch) e atividades de códigos personalizados.

Este implementa a interface IVersoes com as seguintes operações:

Tabela 10: Interface e operações implementadas pelo serviço granular de Documento em Versões.

Interface	**Operações**
IVersoes	AlteraVersaoDocumentoEmUso
	Cancela
	DevolveDocumentoEmProducao
	GuardaDocumento
	GuardaDocumentoAtivoTerminado

4.12 Requerimento e despacho

A lógica associada ao requerimento e despacho implementado no serviço RequerimentoEDespacho.svc, possui igualmente as mesmas operações anteriores do *workflow* do Requerimento e Despacho. Este serviço pode receber um requerimento ou despacho ou retornar um requerimento ou despacho em produção entre outras operações, acabando por implementar 5 interfaces com as suas operações destintas (Tabela 11).

Tabela 11: Interfaces e operações implementadas pelo serviço granular de Requerimento e Despacho.

Interface	**Operações**
IRequerimento	RequerimentoEntregue
IDespacho	DespachoEntregue DevolveRequerimentoSubmetido DespachoSobreRequProcessadoEmFluxoDistinto
IAnaliseDespacho	AnaliseDespachoEntregue DevolveDespachoSubmetido RetornaAccoesDespachante
IReclamacao	ReclamacaoInterposta PresidenteConfirmaDecisao PresidenteNaoConfirmaDecisao PresidenteNaoConfirmaDecisaoParaFim
ISecretariaProcessaRequerimento	SecretariaPodeTramitarRequerimento

A Figura 43 é referente à operação de entrega de requerimento, usando a tecnologia WF. Contudo com a nova abordagem de retirar a lógica de dentro dos *workflows* e também de suprimir a utilização da tecnologia WF, este passou a ser implementado em códigos procedimentais, tais como os outros serviços granulares.

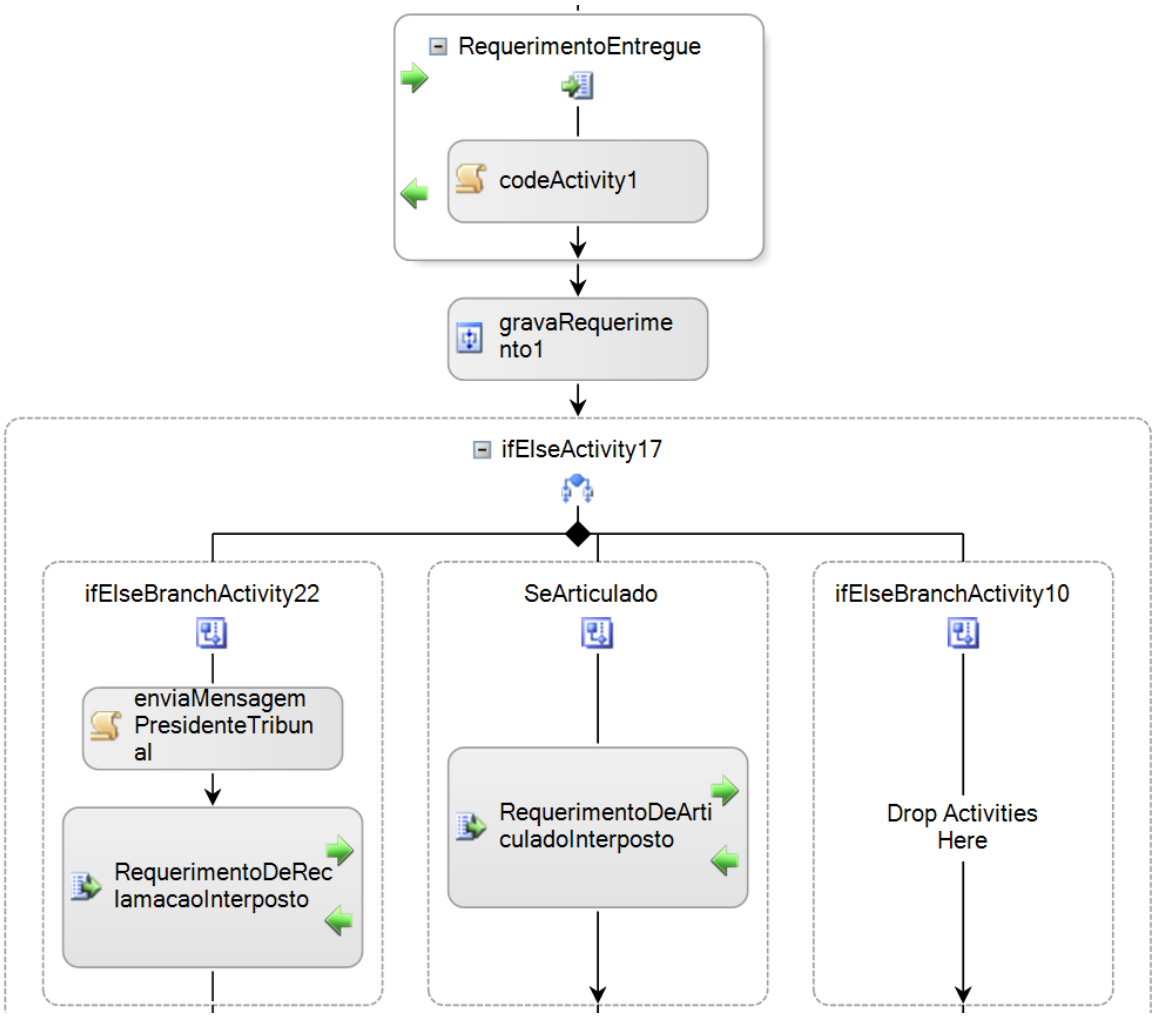


Figura 43: A programação declarativa do evento guardar requerimento entregue, composto pelas atividades pré-concebidas do WF (send e ifelsebranch) e atividades de códigos personalizados.

Como exemplo dessa passagem para códigos procedimentais, temos o trecho de código apresentado anteriormente neste capítulo, que produz o mesmo resultado lógico presente na programação declarativa, mas com abordagens e tecnologias diferentes.

4.13 Inicialização de estados

O serviço de inicialização de estados, pretende colmatar a necessidade que um *workflow* de máquinas de estado possui na entrada de um estado e possa despoletar um evento automático.

Na abordagem anterior, esta necessidade foi preenchida com a utilização da atividade pré-concebido no WF, designado de eventos automáticos (Figura 44). Desta forma, com a nova abordagem, teve-se de arranjar uma

alternativa. A forma encontrada foi criar este serviço REST que utiliza um estilo arquitetural diferente dos demais serviços granulares abordados anteriormente, onde estes foram implementados utilizando SOAP.

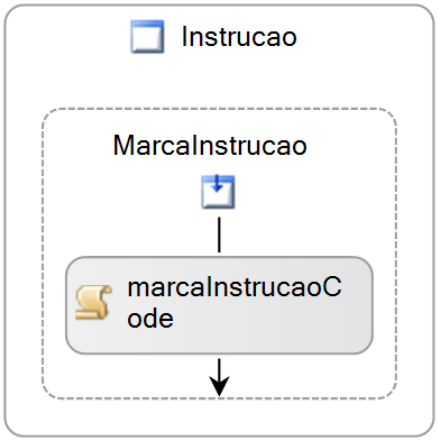


Figura 44: Exemplo da utilização da atividade inicialização automática, quando um determinado processo entra na fase de instrução.

Essa nova forma de ter eventos automáticos, consiste em ter nos modelos visuais dos *workflows* os metadados necessários, ou seja, pontos finais do serviço associados aos referidos estados como um atributo com a descrição "OnEnter". Assim sendo, o motor de *workflows* pode fazer a chamada a estes pontos finais num referido estado constante no modelo, quando entrar num estado que contenha o atributo referido anteriormente com os seus metadados associados.

A lógica por detrás, assenta em receber o identificador da tramitação, o identificador da instância do *workflow* e o tipo de chamada efetuado pelo motor de *workflows*, dando procedimento as instruções do lado da solução do SIJ. Quando terminado com sucesso as instruções do lado da solução do SIJ, o motor é notificado a partir de uma chamada deste serviço, retornando o código de estado de sucesso ou em caso contrário, retornando uma mensagem de erro.

O serviço implementa a interface IStateInicialization contendo as seguintes operações associados a seus pontos finais:

Tabela 12: Pontos finais do serviço granular de Inicialização de estados.

Pontos finais
StateInicialization.svc/AAguardarPrazoReclamacao/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/AAguardarResultadoRecursoExtraordinario/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/AAguardarResultadoRecursoOrdinario/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/AAguardarTransitoEmJulgado/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/AccoesSecretariaRequerimento/{TramitationId}/{WorkflowInstanceId}/{Type}

StateInicialization.svc/AguardaMarcacaoJulgamento/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/AguardarJulgamento/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/Arquivamento/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/ArquivamentoViaMP/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/DesentranhamentoFinal/{TramitationId}/{WorkflowInstanceId}/{Type}
SecretariaExaminaAuto/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/Distribuicao/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/FaseAudienciaContraditoriaPreliminar/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/FaseDeJulgamento/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/Instrucao/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/ProcessoEncerrado/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/ProcessoPenalInitialState/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/Transaccao_AguardarAudiencia/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/Transaccao_FaseDeAudiencia/{TramitationId}/{WorkflowInstanceId}/{Type}
StateInicialization.svc/TransitadoEmJulgado/{TramitationId}/{WorkflowInstanceId}/{Type}

4.14 Configuração dos serviços

Tendo os serviços definidos e implementados, houve a necessidade de efetuar a sua configuração. Desta forma, foram estabelecidos comportamentos no ficheiro de configuração (Código 2), onde é possível definir *tags* WCF, como *behaviors*, que são classes WCF que afetam o funcionamento em tempo de execução do serviço na troca de mensagens entre o cliente e o servidor [69]. Existem vários tipos de comportamentos, sendo *service behavior* (Código 2, linha 2), *binding behavior* (Código 2, linha 15), *contract behavior*, *security behavior* (Código 2, linha 19) *e channel behavior*. [70].

Igualmente, fez-se a configuração das tags *bindings*, onde estes são pilhas de canais pré-configurados, que representam acordos entre um cliente e um servidor [69], sendo especificados o transporte, a codificação e os protocolos envolvidos na comunicação [70].

Os parâmetros desta configuração, advêm dos demais serviços da solução do SIJ. Aconselha-se comportamentos uniformes dos serviços, para que a comunicação entre as partes aconteça sem problemas de maior.

```
<behaviors>
   <serviceBehaviors>
    <behavior name="DebugModeBehavior">
      <dataContractSerializer maxItemsInObjectGraph="2147483647" />
      <serviceDebug includeExceptionDetailInFaults="true" />
      <serviceMetadata httpGetEnabled="true" httpsGetEnabled="true" />
      <serviceCredentials>
        <windowsAuthentication includeWindowsGroups="true"
         allowAnonymousLogons="false" />
      </serviceCredentials>
    </behavior>
   </serviceBehaviors>
</behaviors>
<bindings>
   <wsHttpContextBinding>
    <binding name="LargeMessageBinding" closeTimeout="00:01:00"
      openTimeout="00:01:00" receiveTimeout="00:10:00"
      sendTimeout="00:10:00" bypassProxyOnLocal="false"
      hostNameComparisonMode="StrongWildcard"
      maxBufferPoolSize="250000000" useDefaultWebProxy="true"
      messageEncoding="Text" textEncoding="utf-8"
      maxReceivedMessageSize="250000000" allowCookies="false">
      <readerQuotas maxDepth="32" maxStringContentLength="250000000"
      maxArrayLength="250000000"
      maxBytesPerRead="4096"maxNameTableCharCount="16384" />
      <security mode="None" />
    </binding>
   </wsHttpContextBinding>
</bindings>
```

Código 2: Configuração dos comportamentos e ligações dos serviços no ficheiro de configuração.

Definidos os comportamentos, pôde-se terminar a configuração dos vários serviços. O Código 3 apresenta exemplos da definição de alguns dos serviços granulares.

```
<service
behaviorConfiguration="DebugModeBehavior"name="DespachoRecebido">
  <endpoint contract="MJCVInterfaces.ServiceContract.ServiceWF45
       .IDecisaoMjcvServices" address="" binding="wsHttpContextBinding"
       bindingConfiguration="LargeMessageBinding"
       name="DespachoRecebidoEndPoint"/>
  ...
  <host>
    <baseAddresses>
      <add baseAddress=
       "http://localhost:60277/DespachoRecebido/DespachoRecebido.svc" />
    </baseAddresses>
  </host>
</service>
<service
behaviorConfiguration="DebugModeBehavior"name="SeparaProcesso">
  <endpoint .../>
  <host>
    <baseAddresses>
      <add baseAddress=
       "http://localhost:60277/ProcessoPenalHelper/SeparaProcesso.svc"/>
    </baseAddresses>
  </host>
</service>
```

Código 3: Exemplo da definição de alguns dos serviços no ficheiro de configuração.

Ambos utilizam os mesmos parâmetros nos comportamentos (Código 3, linha 1 e linha 11) e ligações (Código 3, linha 2, 12), mas com diferenças nos contratos (Código 3, linha 2, 12) implementados, nos nomes dos serviços (Código 3, linha 1, 11), pontos finais (Código 3, linha 2, 12) e endereços base (Código 3, linha 5, 14).

4.15 Resumo

Este capítulo focou-se em especificar as diretrizes da criação dos serviços granulares. Primeiramente apresentou a arquitetura modular, onde a definição se encontra na camada Vista e a implementação na camada Lógica. Procurou-se fazer a contextualização dos componentes WF que serviram de base para a reengenharia que deram origem a esses serviços. Esses componentes podem ser diferentes tipos de atividades que compõem uma programação declarativa, destacando-se por serem discretos e reutilizáveis, desenhados para cumprir um propósito definido.

Fez-se igualmente a apresentação de cada serviço granular, as suas interfaces e as operações que os compõem, assim como, a lógica conceptual, em figuras de programação declarativa de WF, associado a cada um desses serviços. Para finalizar o capítulo, abordou-se as configurações realizadas para que estes serviços estejam disponíveis para serem acedidos. Apresentou-se as configurações dos comportamentos e ligações destes elementos necessários para interagir com os demais componentes já existentes na solução.

5. A nova abordagem

No capítulo anterior apresentou-se as diretrizes da criação dos serviços granulares. Este capítulo vai mais além, mostrando como esses serviços se enquadram na nova arquitetura do sistema. Primeiramente será abordada a integração da ferramenta de modelação, o editor de metadados, padrões definidos para os modelos e a ferramenta de definição de processos. Com a nova notação estabelecida, os *workflows* começaram a ser modelados com base neste novo padrão. Assim, é igualmente pertinente apresentar o *workflow* do processo penal com base nessa nova notação. Seguidamente, é abordada a constituição da API do motor de *workflows*, que é o centro das operações do sistema de *workflows*.

Para finalizar o capítulo, são apresentadas as alterações sofridas nos componentes já existentes na solução e a execução de testes sobre essas alterações. Apresenta-se ainda o novo componente, o serviço compositor que advém da necessidade de centralizar as requisições feitas e orquestrar a pedidos dos vários componentes da solução do SIJ a API.

5.1 Modelação de workflows

Ao contrário do que acontecia anteriormente, onde o modelo visual era modelado diretamente no WF a partir da ferramenta de modelação embutido no framework, nessa nova abordagem, a criação do modelo visual realiza-se utilizando a ferramenta *what you see is what you get* (WYSIWYG) de modelação Microsoft Visio. Este, face as características que possui, permite que depois de modelados os *workflows*, estes possam ser posteriormente convertidos para o formato XML, o qual, a API do motor de *workflows* utiliza para realizar as suas operações.

Com o objetivo de ter uma coerência na definição dos objetos na fase de modelação, a equipa de desenvolvimento reuniu para definir uma notação que delineasse a associação de cada objeto a fazer parte da modelação a uma consequente entidade no sistema judicial.

Tendo a notação definida e a ferramenta escolhida para suportar a modelação, sentiu-se a necessidade de ter mais duas ferramentas que auxiliassem no processo de modelação e na definição de processos, que fosse compreensível para o motor de *workflows*.

Desta forma, surge o conversor WYSIWYG, projeto de investigação realizado na mesma equipa de trabalho no Instituto de Engenharia Electrónica e Telemática de Aveiro (IEETA). O funcionamento do conversor consiste basicamente em extrair o ficheiro *.vsdx (Microsoft Visio Diagram XML)*, formato baseado em XML e que usa a estrutura do ficheiro .ZIP para juntar todos os conteúdos do desenho, renomeando a sua extensão para formato .zip dando origem a metadados, imagens, ficheiros textos e pastas referentes ao ficheiro extraído [71]. Os ficheiros extraídos são programaticamente analisadas e recolhidas informações necessárias para construir o ficheiro XML que vai constar do repositório de *workflows* a ser consumido pelo motor (também desenvolvido pela equipa, como referido no primeiro capítulo). Na terminologia da automatização de processos

essa ferramenta pode ser comparada ou descrita como sendo ferramenta de definições de processos, que encontra no linear entre as áreas funcionais do tempo de construção e do tempo de execução (Figura 45).

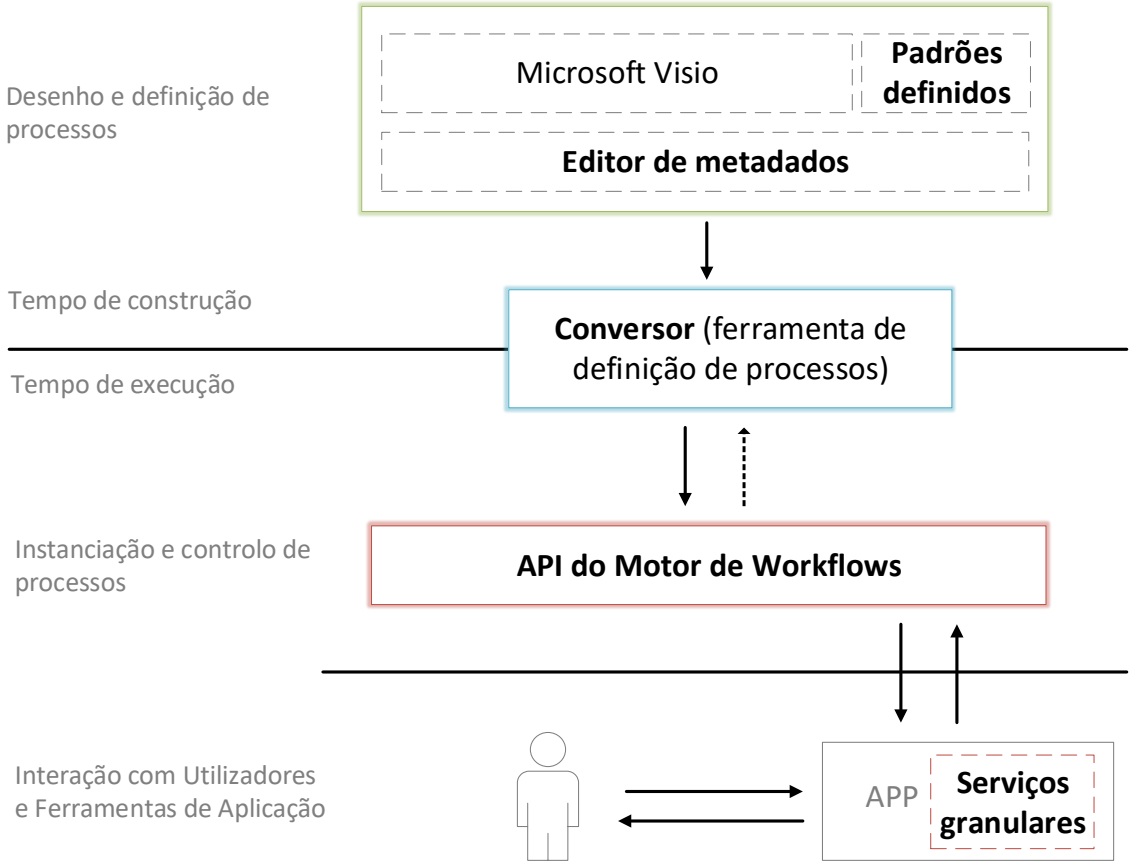


Figura 45: Modelo conceptual da integração das ferramentas de modelação, editor de metadados, padrões definidos para os modelos, ferramenta conversor da definição de processos e a API do motor de workflows [32].

A outra ferramenta é o editor de metadados, este é responsável por associar a cada objeto a ser desenhado no modelo visual propriedades, classes, pontos finais e outros metadados a estes. Assim será possível ao motor enviar estas informações para o sistema quando este as requisitar.

Suponha-se, por exemplo, a necessidade de representar que numa transição de A para B deve ser executado uma determinada operação de serviço, ou apresentado um determinado formulário na interface da aplicação cliente. A atribuição dos metadados aos modelos visuais, numa primeira fase estava a ser feito a partir da Microsoft Viso, mas constatou-se que este, apesar de funcionar, não seria a forma mais eficiente de continuar a efetuar esta tarefa. Pois para cada metadado a introduzir, teria de ser inserido manualmente, editando objeto a objeto.

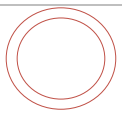
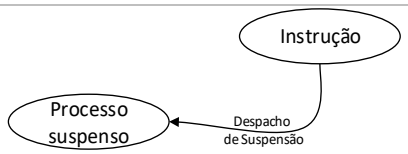
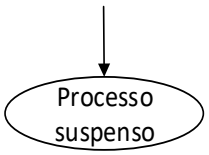
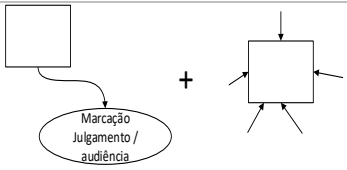
Face a isso, a equipa construiu esse editor que importa o modelo visual com os objetos modelados para a ferramenta e lista todas as transições possíveis e todos os estados definidos neste modelo. Com os estados e transições listados de forma hierárquica pode-se associar a cada um desses objetos os metadados correspondentes, facilitando a tarefa aos analistas e programadores. Ainda a esta ferramenta foi adicionada a

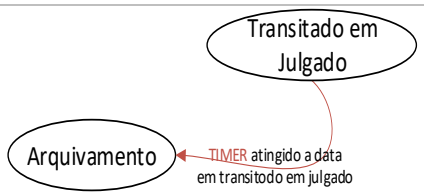
funcionalidade de aceder a solução do sistema e listar todas as classes e pontos finais dos serviços que constam no SIJ, podendo estes ser associados a um objeto, bastando a sua seleção na lista retribuída da consulta.

5.1.1 Notação estabelecida

Era pertinente fazer a contextualização dos objetos e sua respetiva descrição para que o processo de modelação tornasse coerente. Este consistiu no levantamento dos diferentes conceitos e termos tal como abordados na secção 2.4.1 e a definição da forma geométrica dos objetos que serão associados a inicialização de um *workflow*, estados, transições até a forma de representar os sub-*workflows* (Tabela 13). A base desta nova notação são objetos UML. Importa também ter em consideração, que esta notação deve ser de perceção fácil para os elementos da comissão de acompanhamento (Juízes, Procuradores, Advogados e Oficiais de Justiça).

Tabela 13: Notação estabelecida para a modelação dos workflows.

Elemento	**Descrição**	**Notação**
Início	Círculo	
Fim	Círculo duplo	
Estado	Elipse	
Eventos em estados (*onEnter*, *onLeave*)	Metadata (*onEnter*, *onLeave*)	Sem notação
Transição "manual" (despoletados por eventos efetuados por utilizadores)	Seta	
Transições diretas (para estado específico independente do estado atual)	Seta (sem estado de início) só com estado final	
Eventos (eventos que não despoletam transições) comuns a vários estados	Quadrados ligados aos estados que utilizarem as coisas comuns	

Sub-Workflows	Círculo	
Temporizador	Seta	

5.1.2 Workflow do processo penal com base na nova notação

Com a notação estabelecida, tornou-se possível representar os vários conceitos associados aos seus objetos e respetivos metadados no *workflow* do processo penal. Sendo assim é possível constatar na Figura 46, na área A, a definição dos metadados de um estado de *workflow* que possui transições automáticas. Na área B desta mesma figura, tem-se a representação do estado inicial de um determinado *workflow* e na área C tem-se o objeto elipse que representa um determinado estado. Cada estado deve conter um identificador único.

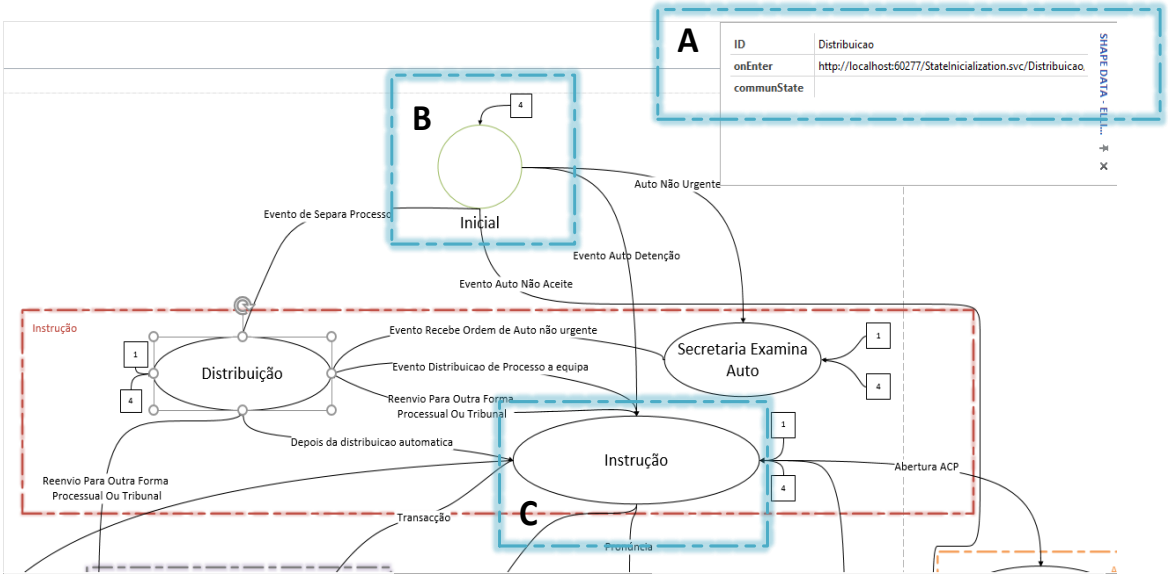


Figura 46: Definição de estados e de mecanismo para eventos automáticos.

Na Figura 47, na área A, tem-se definido os metadados referentes a uma transição manual (despoletado por ação humana). Numa transição manual é definida a operação do serviço granular associado a essa transição, o *namespace* do objeto a ser passado como parâmetro na operação do serviço e a secção de cargos que tem permissão de executar essa transição. Igualmente nesta mesma figura, na área B, pode-se constatar a representação gráfica de uma transição manual. Na área C tem-se a representação de eventos comuns a vários estados, mas que não despoletam transições de estado, ao contrário das transições manuais e automáticas.

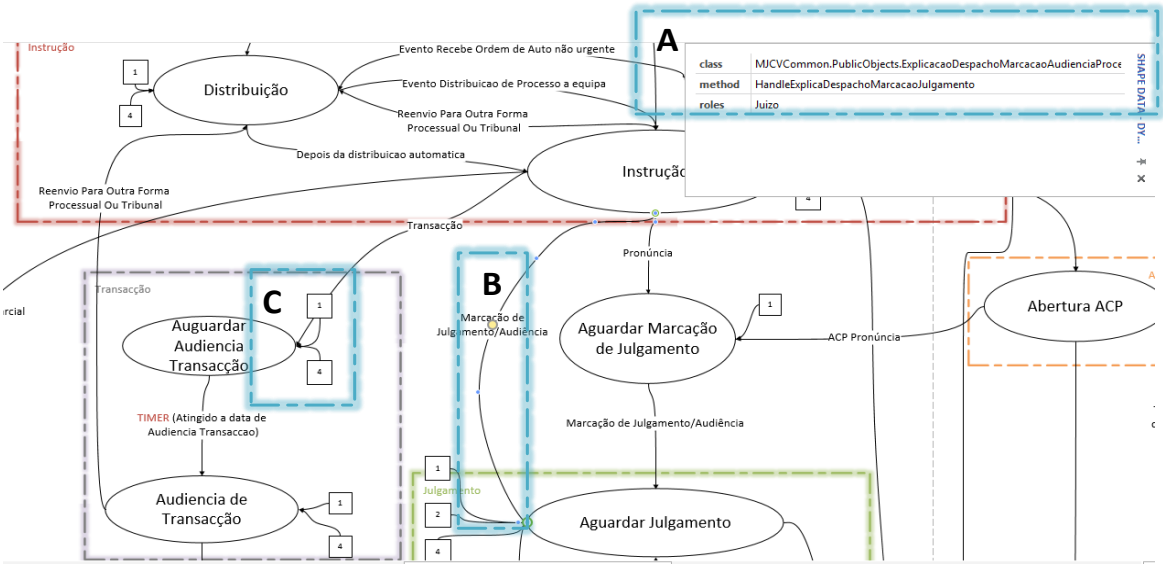


Figura 47: Transições e definição dos metadados.

Os eventos comuns foram separados em quatro representações diferentes. Na notação estabelecida, este é representado pelo quadrado com as setas interligados, sendo a segunda parte da representação de eventos comuns. A segunda parte da representação de eventos comuns é colocado num separador diferente do ficheiro do *workflow* do processo penal, mas este continua associado ao ficheiro principal graças ao identificador único que cada quadrado possui. Com os quadrados pequenos associados aos estados pode-se manter-se a relação desses eventos definidos no ficheiro a parte, tal como apresentado na área C da figura anterior. Quando um estado tiver associado a si um ou mais desses quadrados maiores, significa que é possível nesta fase do processo emitir decisões como alteração e rol, pedidos de diligências, alteração de acusação entre outras (Figura 48).

Figura 48: Representação dos eventos comuns.

Igualmente é possível a entrada de decisões, como esclarecimento, alteração de sentença, pedido de advogado oficioso, apensação entre outras constantes da Figura 49. As descrições 1, 2, 3, e 4 nesses quadros são meramente ilustrativos, a fim de diferenciar estas decisões/explicações que podem ser efetuados na mesma fase.

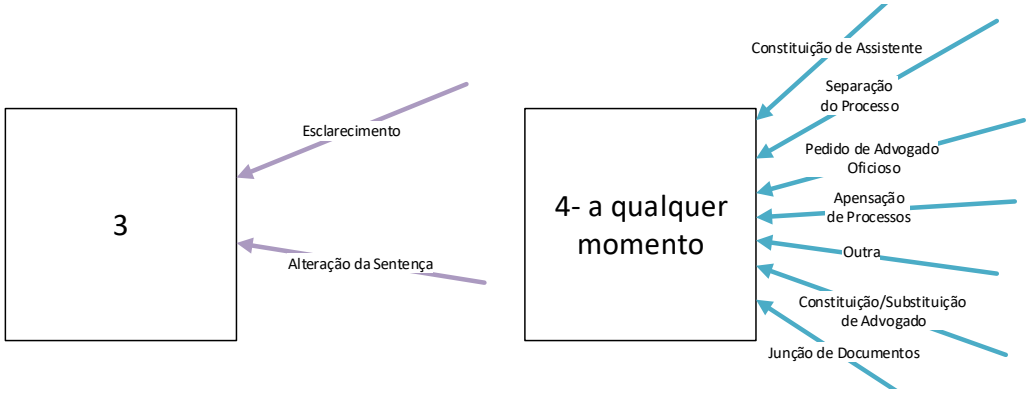


Figura 49: Representação dos eventos comuns, continuação, parte II.

Outra representação estabelecida na notação, são os temporizadores. A representação dos temporizadores assenta em ter nos metadados de uma transição despoletada pelo temporizador, o ponto final do serviço a ser chamado quando atingida a data da tramitação (área A, Figura 50). Visualmente, as transições que dependem de temporizadores são marcadas com a *tag* TIMER mais a descrição da transição (área B, Figura 50).

Por fim, ainda nesta figura, tem-se a representação do estado "final de um *workflow*". Este é representado por dois círculos (área C, Figura 50) contendo nos metadados, as mesmas propriedades de um estado representado por elipse.

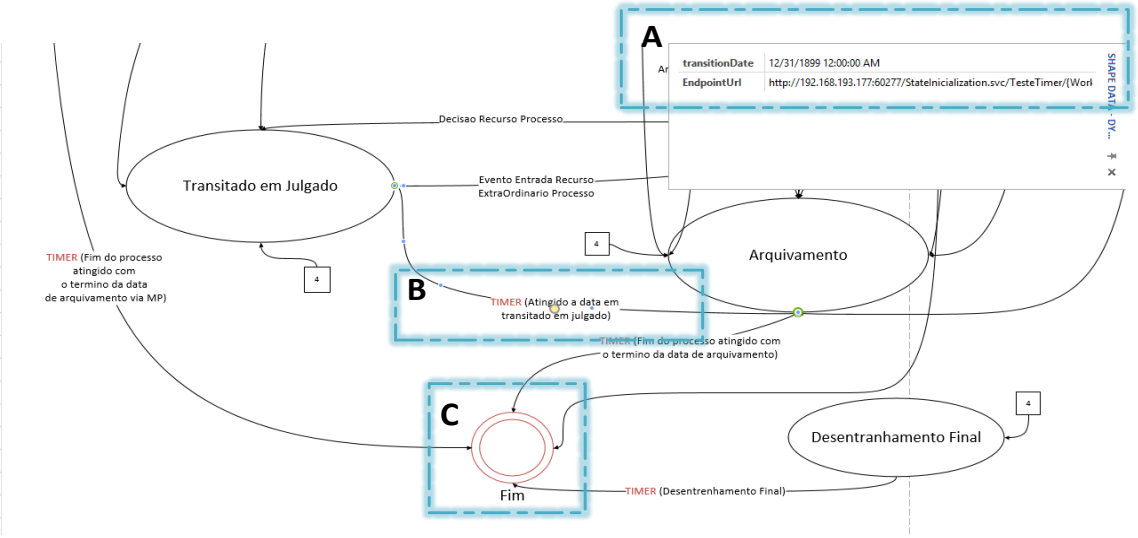


Figura 50: Temporizador e a definição do estado final.

5.2 API do motor de workflows

O motor de *workflows* foi desenvolvido em Python e congrega uma API que lhe permite a utilização por várias aplicações distintas em simultâneo. Foi desenvolvido com a framework Django (framework de desenvolvimento escrito em Python que utiliza o padrão arquitetural Model-View [72]). A API do motor de *workflows* consiste na disponibilização de pontos finais a serem consumidos pela solução do SIJ (ou outra qualquer aplicação que necessite de integrar *workflows* na sua lógica). Possui funcionalidades que foram

construídas com base nas necessidades de um sistema de *workflows*, conjugadas com as necessidades próprias da realidade de fluxos da área da justiça.

O sistema de *workflows* é composto por um motor que é o centro das operações, uma base de dados objeto-relacional (ORDBMS) PostgreSql [73] para a persistência dos metadados dos *workflows* (Figura 51) e é acedido via uma API. É parte integrante deste um repositório onde constam os modelos visuais convertidos em ficheiros XML a serem consumidos pelo motor. Desta forma, a arquitetura da interface vai de encontro com o modelo de referência proposto pela WfMC abordado na secção 2.5.

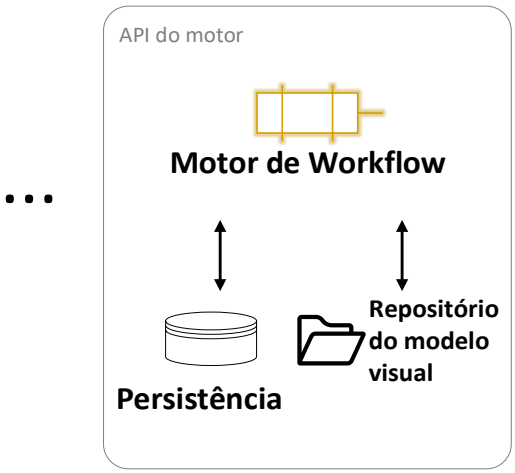


Figura 51: Composição do motor de workflows.

A partir desta API é possível criar uma instância de *workflow*, pode-se tramitar um *workflow*, pode-se consultar a lista de transições de uma instância num determinado estado além de outras funcionalidades descritas detalhadamente na Tabela 14.

Sendo uma API que expõe para o exterior as interfaces, este tem de utilizar a rede para comunicar. Essa comunicação é feita a partir da infraestrutura HTTP/HTTPS, utilizando o método GET para solicitar dados de um recurso específico. O método GET permite que os pedidos fiquem em cache, mas por outro lado restringe o tamanho do pedido a ser feito, sendo aconselhável a sua utilização somente para a recuperação dos dados [74].

Além da utilização do método GET, a API também utiliza o método POST. A partir do método POST é possível submeter os dados a serem processados para um recurso específico, em contrapartida, este não permite que os dados fiquem em cache [74].

Tabela 14: Pontos finais da API do motor de workflows.

Métodos	**Pontos finais**	**Descrição**
GET	*/api/instances/{id_instance}*	A partir da instância, este retorna o estado e nome do *workflow* associado.

GET	*/api/instances/{id_instance}/currentState*	Retorna o estado atual do *workflow* mediante a instância.
GET	*/api/instances/{id_instance}/transitions*	Lista todas as transições possíveis de uma instância.
GET	*/api/wf/{workflow}*	Mediante o nome do ficheiro do *workflow* constantes no repositório, este retorna o nome do ficheiro e o título do *workflow* que pode ser diferente do nome do ficheiro.
GET	*/api/wfs/{active}*	Este retorna o título e o id de todos os tipos de *workflows* que se encontram ativos e inativos.
GET	*/api/wfs/{workflow}/states*	Retorna o título referente a cada estado de uma instância de *workflow*.
GET	*/api/wfs/{workflow}/{src}/{dst}/roles*	Recebe os parâmetros, nome do fluxo a origem e o destino e retornando todos os cargos que podem intervir numa transição que cumpra esses requisitos.
GET	*/api/wfs/{workflow}/{state}/transitions*	Lista todas as transições possíveis num determinado estado de um *workflow*.
POST	*/api/instances/{id_instance}/changeState/{state_dest}*	Força a tramitação de uma instância de *workflow* para um estado específico desejado.
POST	*/api/instances/{id_instance}/overrideState/{state_override}*	Este é útil para sobrepor um estado de uma instância de *workflow*. Suponha ter-se a necessidade de voltar para um estado anterior, pode-se utilizar esta interface para retornar a um estado específico.
POST	*/api/wfs/{workflow}/start*	Permite criar uma nova instância de um determinado *workflow*, ou seja, iniciar um novo *workflow*.

Ainda, mediante a lógica inerente ao funcionamento de um tipo de *workflow*, é possível invocar uma nova chamada. Esta pode acontecer quando atingindo um determinado prazo (temporizador) numa instância, aciona-se um gatilho a partir do motor que faz uma chamada a um determinado ponto final dos serviços do SIJ (secção 2.4.1), mediante os metadados constantes do modelo visual.

5.3 A nova arquitetura do sistema

Face as adaptações realizadas, a arquitetura do sistema sofreu algumas alterações, tendo sido inseridos novos componentes. Além da inserção de novos componentes, também foram removidos outros que constavam da arquitetura anterior apresentado na secção 3.2, advindo da mudança de paradigma. A alteração mais marcante aconteceu com a supressão do WF que era responsável pelos serviços de *workflow*. Toda a lógica de negócios que estava na responsabilidade dos serviços de *workflow* foi transformada em serviços granulares, usando o protocolo SOAP para a comunicação. Estes serviços granulares nos *workflows* anteriores eram componentes designados de atividades de código e atividades WF, estando agora localizados na camada Vista (Figura 52), ao contrário dos anteriores que estavam definidos na camada Lógica. Esta alteração justifica-se com a necessidade de modularizar o acesso à camada lógica, usando WS como interface que centraliza as requisições. Contudo, face a natureza deste serviço, este requer a implantação num servidor web ao contrário de serviços Windows (que permitem criar aplicações executáveis de longa duração executados em sessões do Windows [75]).

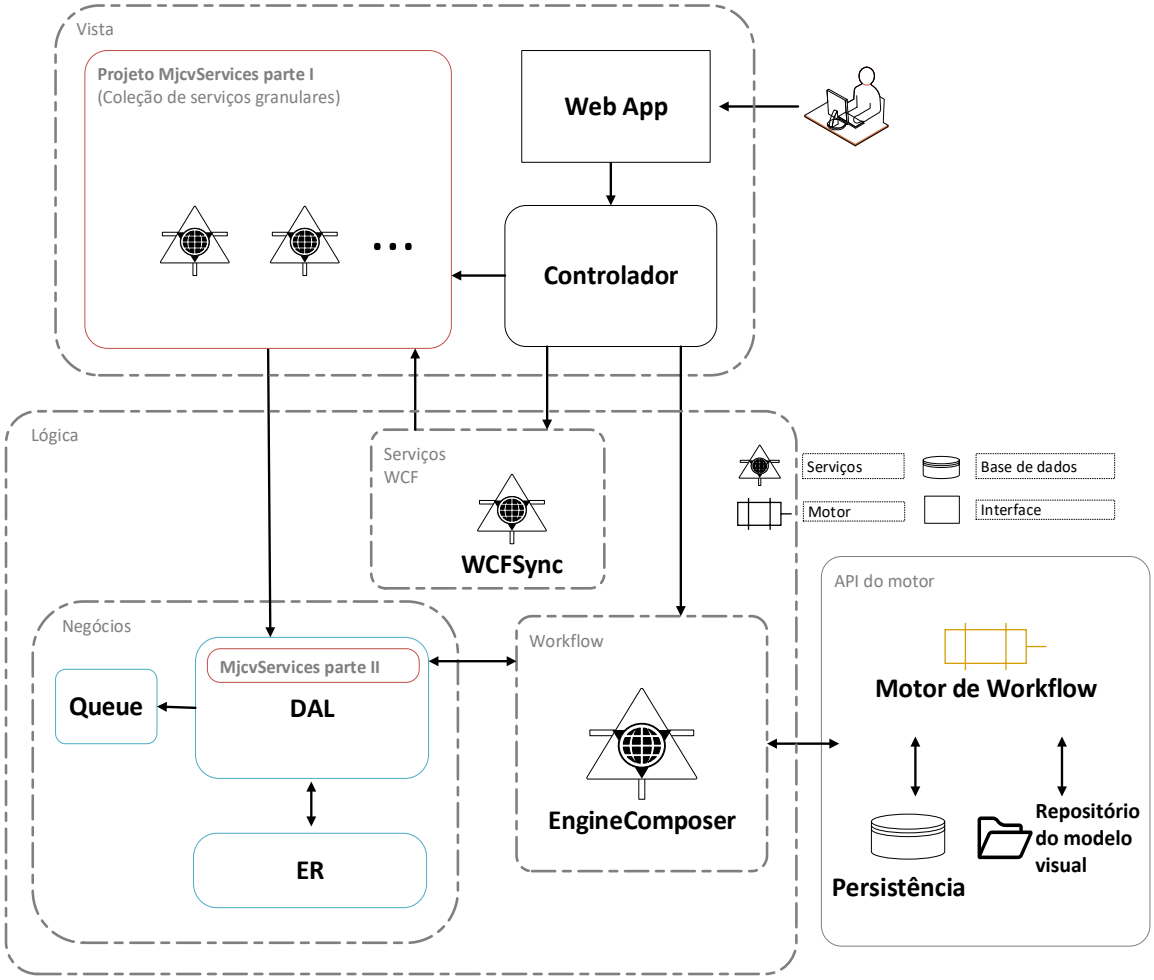


Figura 52: A nova arquitetura conceptual do sistema.

Outra alteração efetuada foi a inserção da API do motor de *workflows*, que passou a fazer as operações que eram da responsabilidade do WF. A API é constituída pelo repositório de modelos visuais e uma base de dados

de persistência para os metadados dos *workflows* em execução tal como descrito na secção 5.2. O motor é o componente central que gere todas a requisições, despoleta as transições e permite comunicar com o exterior.

Ainda nesta nova arquitetura, permanece o serviço de comunicações WCFSync que continua a desempenhar as mesmas funções. Mas por outro lado, foi adicionado a subcamada designado de Workflow um novo serviço compositor EngineComposer. O serviço compositor é responsável por interligar a solução do SIJ com a API do motor de *workflows*, fazendo a requisição às interfaces abordadas na secção 5.2 da API, retornando a resposta para a solução principal do SIJ. Este serviço comunica com a API usando a infraestrutura HTTP (REST), e com os demais componentes da solução onde este encontra definido, a partir do protocolo SOAP.

5.3.1 Serviços auxiliares

Ao contrário da arquitetura anterior, onde existem os dois serviços utilitários para o auxílio dos serviços de *workflows* em operações como: permitir a criação da instância do *workflow* mediante o tipo, a conversão do *workflow* antigo, a mudança de estado, abortar o *workflow* em execução, forçar o descarregar do *workflow* da memória para a persistência, listar os diversos tipos de *workflows* que constam na base de dados de persistência, as possíveis explicações de despachos mediante o estado de cada instância ou a listagem dos diferentes estados de um determinado *workflow*. Nesta nova arquitetura tudo isto está a ser feito diretamente na API do motor de *workflows* (Figura 53), acabado por serem supridos esses serviços auxiliares.

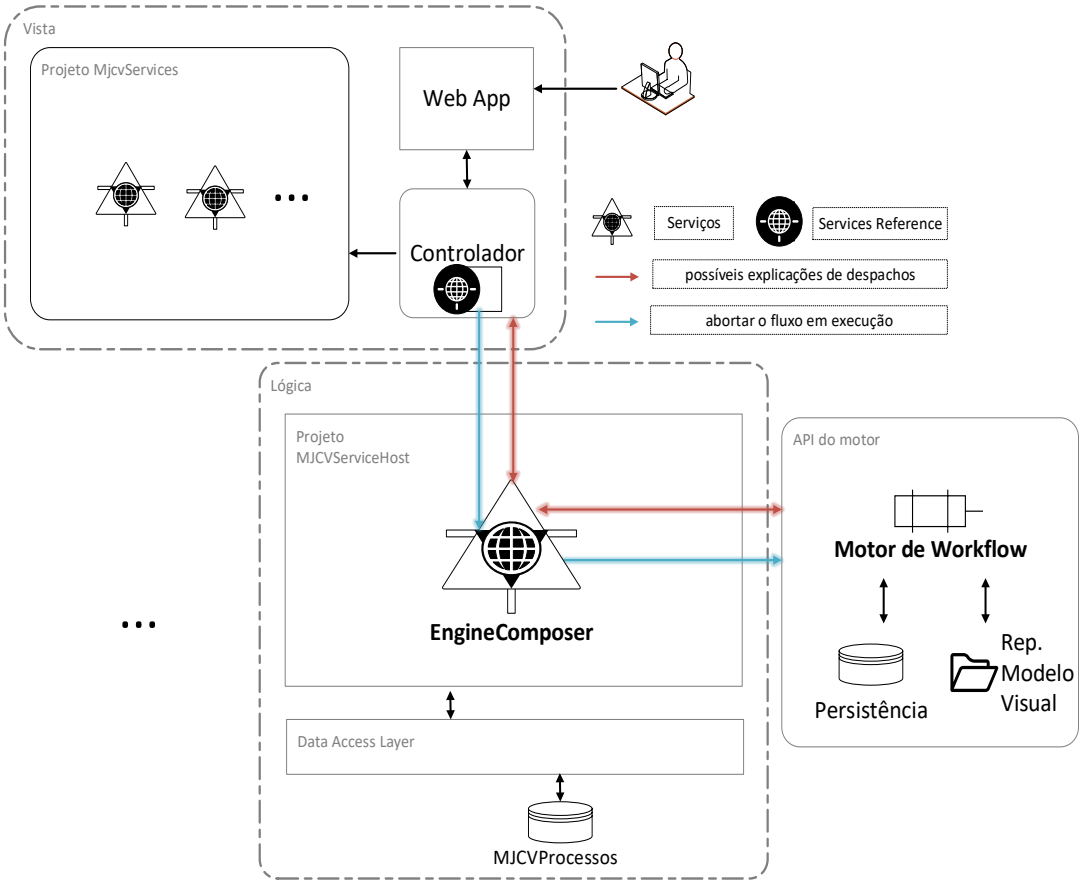


Figura 53: A nova arquitetura conceptual da interação dos serviços em tarefas como "possíveis explicações de despachos" ou "abortar o workflow em execução", independente do tipo de workflow.

Com a nova abordagem, os antigos serviços utilitários referidos em supra, acabaram por ser substituídos por um único serviço, o serviço compositor, que requisita as informações necessárias à API do motor de *workflows* onde estão implementadas as interfaces que disponibilizam estes dados.

Estando todas estas operações centralizadas, torna-se mais fácil a manutenção do código por parte da equipa, possibilitando a inserção ou remoção de funcionalidades.

Outra alteração na arquitetura, com adoção desta nova abordagem, tem a ver com a interação dos diferentes serviços na instanciação de um *workflow* de processo penal ou no despertar de alguns eventos no sistema judicial, como por exemplo a interposição de um despacho ou requerimento. Desta forma, a Figura 54 apresenta essa mudança arquitetural e de componentes em comparação com a que foi abordado na secção 3.2.1. Suponha-se então que foi interposto um despacho. Este inicia-se a partir da interface da instância web tal como o anterior, chegando ao controlador. A partir do controlador é feito a referência a uma operação qualquer do serviço granular de Documentos em Versões, definido na camada Vista onde estão os serviços granulares. Ao contrário do que acontecia antes, onde este era implementado como serviço WF.

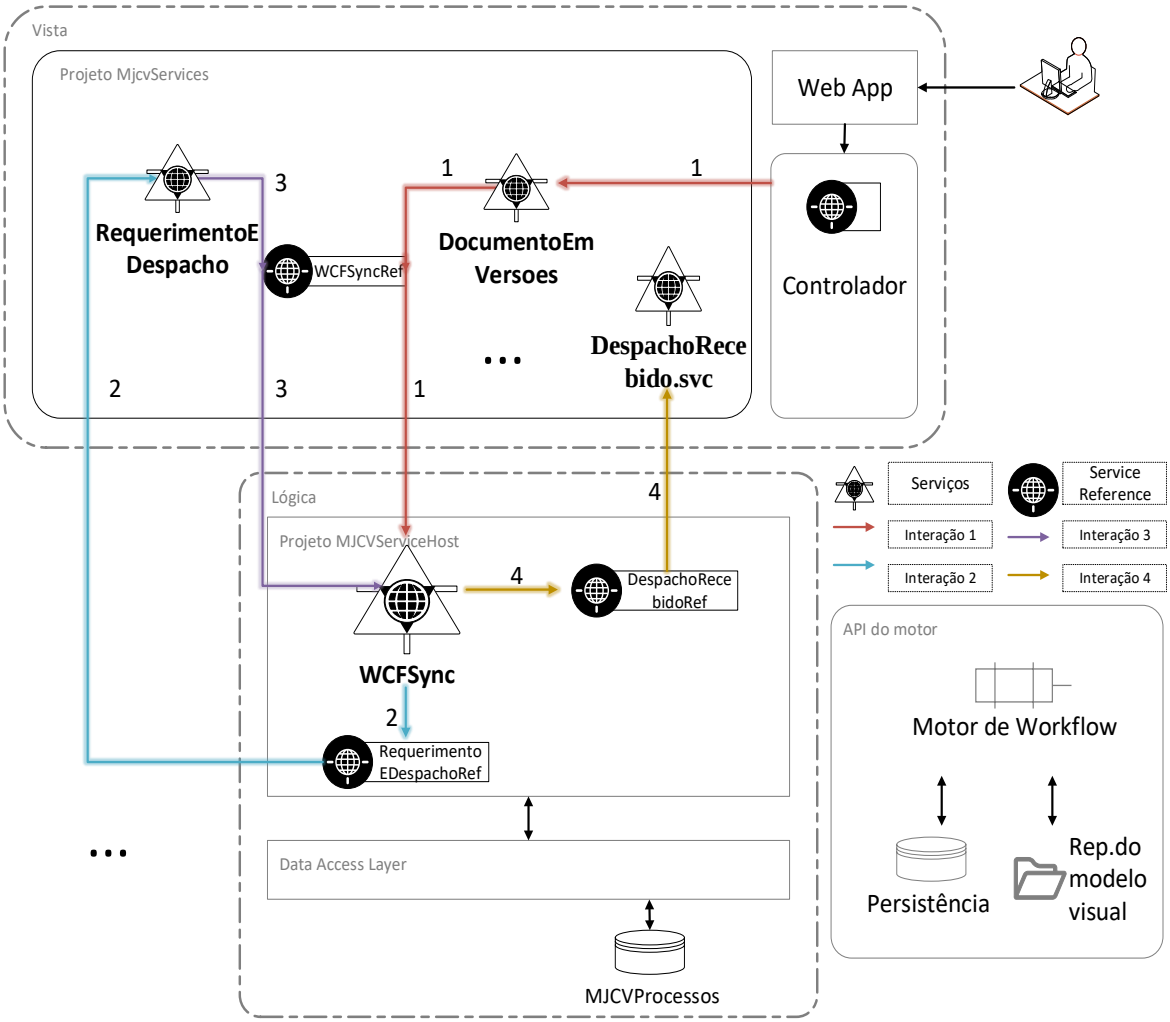


Figura 54: A nova arquitetura conceptual da interação dos diversos serviços a quando da entrada de um despacho ou requerimento para o processo penal.

Após o documento ter sido dado como terminado o serviço granular de Documentos em Versões faz referência ao serviço de comunicação que continua a ser definido e implementado no projeto de livraria de classes MJCVHosting e executado na instância MJCVServiceHost. Este serviço, após receber o documento terminado envia-o para uma operação implementada pelo serviço granular de Requerimento e Despacho (a interação numero 2), que tal como o serviço granular de Documentos em Versões está agora definido na camada Vista.

Tendo o despacho sido finalizado, o serviço granular de Requerimento e Despacho faz referência ao serviço de comunicação para que este possa dar continuidade (a interação número 3). Mediante o tipo de documento proferido, o serviço de comunicação invoca uma das várias operações implementadas pelos vários serviços granulares na camada Vista, que antes estavam implementados num único serviço WF do *workflow* do processo penal (interação número 4), continuando a interação com os novos componentes da arquitetura (API do motor de *workflows* e serviço compositor).

Desta forma, deixa-se de ter um único serviço de *workflow* de processo penal para ter vários outros pequenos serviços granulares, permitindo maior reutilização de código e maior abertura para outros serviços e componentes que precisem destas operações, pois antes eram apenas métodos que não eram possíveis de ser acedidos diretamente como serviços. Com essa transformação para serviços, será possível aceder às suas interfaces mediante o cumprimento de requisitos e privilégios.

5.4 Adaptações no sistema de workflows

Nas secções anteriores deste capítulo, foram apresentados de forma conceptual a nova arquitetura, os componentes atuais e fez-se referência aos que já não fazem parte da arquitetura. Foram igualmente apresentadas a notação estabelecida para a modelação dos modelos visuais, o conversor para a definição do processo em ficheiro XML a ser utilizado pela API bem como o editor de metadados. Por fim, falta apresentar as alterações específicas de cada projeto da solução que foram realizadas neste estudo.

Desta feita, nesta secção apresenta-se a criação do serviço compositor para orquestrar a comunicação entre a API do motor com a solução do SIJ e por fim algumas alterações efetuadas nos projetos existentes.

5.4.1 Serviço compositor

O propósito do serviço compositor advém da necessidade de centralizar as requisições feitas à API do motor de *workflows*, com ganhos sobretudo na reutilização. Este além de orquestrar a requisição dos vários componentes da solução do SIJ com a API, também possui métodos e classes auxiliares que tratam os dados, fazendo a serialização e deserialização de objetos JavaScript Object Notation (JSON serializador de dados e formato de mensagens [76]) retornados da API do motor de *workflows*, ou instanciando classes de objetos e preenchendo-os de acordo com o esperado pelo componente que fez a solicitação.

Sabendo de antemão que se trata de um trabalho de adaptação, é importante levar em consideração componentes já existentes, que aguardam os dados numa determinada estrutura e formato. Desta forma, quanto mais cirúrgicas e cuidadas forem as alterações menos probabilidade haverá de *side effects* (refere-se à modificação de algum tipo de estado no código [77]) ao longo da utilização do sistema.

Assim sendo, no trecho da arquitetura conceptual do sistema (Figura 55), o compositor aparece entre a camada de negócios e a API do motor de *workflows*, com a responsabilidade de fazer a ponte entre o SIJ e a API. Tecnicamente o serviço implementa uma única interface IWorkflowEngineComposerService, sendo definidas diversas operações (Tabela 15).

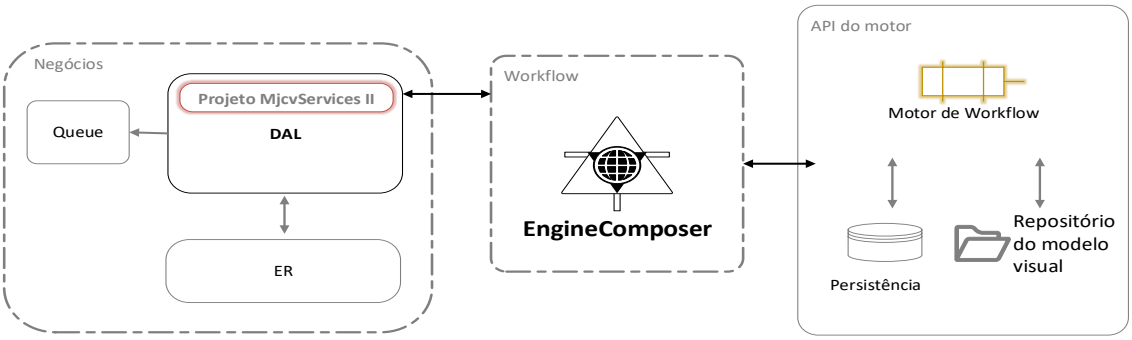


Figura 55: Arquitetura conceptual da interação do serviço compositor com outros componentes do sistema.

Tabela 15: Descrição das operações do serviço compositor.

Operações de contrato	**Descrição**
Ative	Recupera todos os *workflows* ativos.
ChangeState	Permite a um componente da solução do SIJ tramitar uma instância de *workflow* na API motor.
CurrentState	Recupera o estado atual de uma instância de *workflow*.
GetAvailableOps	Retorna um objeto com as informações como explicações de despachos possíveis, transições e outras informações de um tipo de workflow mediante uma instância.
OverrideState	Permite a um componente da solução do SIJ voltar a um estado anterior de uma instância.
SourceDestRoles	Recupera os cargos que podem fazer tramitar uma instância.
Start	Permite a um componente da solução do SIJ criar um tipo de *workflow*.
States	Recupera todos os estados que um tipo de uma instância de *workflow*.
Title	Recupera o título de uma instância.
TransitionDate	Recebe a data e a instância de workflow para ser processado as transições automáticas no motor de workflows.
Transitions	Recupera todas as transições possíveis de uma instância de *workflow* no estado em que se encontra.

TransitionsByState	Recupera todas as transições possíveis de uma instância num determinado estado.
Type	Recupera o tipo de uma instância de *workflow*.

A API interage indiretamente com as bases de dados do sistema. A base de dados do SIJ e a base de dados do sistema de *workflows* anterior, utilizam o tipo de dados *uniqueidentifier* (GUID) assente em 16 bytes [78], para as instâncias das tabelas associados. Ao contrário dos anteriores, a API do motor de *workflows* utiliza *Identity* (INT). Suponha-se um processo que tenha associado a si um *workflow* que na persistência do motor tem uma das instâncias representado por 1, enquanto que este mesmo valor é representado na base de dados do SIJ por 00000000-0000-0000-0000-000000000001 no formato GUID.

Desta forma, surge um novo problema a resolver nesta adaptação. Para resolver esta situação, criou-se uma classe auxiliar para converter INT para GUID (levando em consideração a diferença de tamanho entre estes) e vice-versa, permitindo a troca de dados entre o SIJ e a API. Mas mesmo assim, por estar-se a utilizar o método HTTP GET em requisições ao API, precisou-se também de converter do tipo INT para STRING, para ser possível enviar os dados a partir do cabeçalho, visto que o método GET só permite *array* de STRING no cabeçalho de uma requisição.

Contudo, estas comunicações a partir da infraestrutura HTTP tem por vezes os seus contras. É importante que ao fazer uso destas tecnologias, tenha-se formas e técnicas para manter a consistência, uma baixa latência e saber lidar com a concorrência, segurança e confidencialidade da informação.

Sendo assim, torna-se importante garantir a integridade e a disponibilidade da comunicação entre a solução do SIJ e a API do motor de *workflows*, uma vez que os dados estão a ser solicitados a partir da rede com limites no I/O.

A integridade dos dados num serviço REST pode ser conseguida utilizando algumas técnicas, tais como HTTP Básico (o cliente envia o nome do utilizador e a palavra passe como texto não cifrado, pelo que se aconselha a utilização com HTTPS [79]), HTTP Digest (o cliente envia a palavra passe em forma de *hash* [79]), uso de Certificados, autorização baseado em *Tokens*, bem como, o padrão aberto OAuth. Mas devido a visibilidade destes serviços serem somente dentro da organização, nenhum dos mecanismos se encontra atualmente implementado. Apesar de não serem comtemplados com nenhum desses mecanismos específicos de segurança, sabe-se de antemão que estes serviços são protegidos pela infraestrutura onde se encontram alojado.

Não obstante, pode-se futuramente adotar a autenticação baseada em *tokens* (implementação própria ou utilizar outras implementações genéricas tais como JSON Web Tokens [80], secção 2.7). Este mecanismo é conhecido por ser *stateless*, não sendo necessário guardar sessões, estando as informações necessárias para a autenticação na URL a ser transmitidos. Não está associado a um esquema de autenticação específico. Garante um melhor desempenho comparando com outros tipos de mecanismos, onde gerar um *hash* simples pode levar menos tempo do que uma consulta a uma base de dados relacional [55].

Por outro lado, para garantir a disponibilidade do serviço, no desenvolvimento acabou-se por utilizar a técnica de programação assíncrona (Código 4), que permite evitar estrangulamentos e ajuda na melhoria da capacidade de resposta geral do serviço.

A utilização da programação assíncrona é essencial para acesso a um recurso utilizando uma rede mais lenta ou em ambientes computacionais exigentes. Suponha-se uma atividade ao aceder ao API onde utiliza o modo de gestão de instância única (cria-se apenas uma instância do serviço fornecedor para responder a todas as requisições dos serviços clientes) e for bloqueada dentro de um processo síncrono, todo o serviço fica bloqueado a aguardar. Por outro lado, em um processo assíncrono, o serviço continua com as tarefas que não dependem do recurso da rede até a tarefa que se encontra bloqueada termine [81]. Este tem como centro da sua programação objetos *Task* e *Task<T>*, que modelam as operações assíncronas, sendo suportados pelos operadores *async* e *await* [82]. *Tasks* são construções usadas para implementar o que é conhecido como o Modelo Prometido de Concorrência. Este modelo oferece uma "promessa" de que o trabalho será concluído em um ponto mais à frente, possibilitando a continuidade das outras tarefas em sequência no método que fez a requisição [83]. Pode ser conseguido com a atribuição do operador *await* a uma chamada, acabando este por permitir a execução do trabalho útil enquanto uma tarefa está sendo executada, retornando o controlo para o método que fez a chamada.

Além da necessidade da utilização da programação assíncrona, constatou-se que ao aceder a uma API recebendo diversos tipos de coleção de dados, torna-se necessário o uso de algum mecanismo que de uma forma ou de outra auxilie a escrever métodos que possam ser reutilizados por todas as coleções de dados. Desta forma adota-se a técnica de tipos genéricos, sendo este também, parte do .Net.

Os tipos genéricos permitem a definição de estruturas de dados tipo seguro, sem se comprometer com os tipos de dados reais [84], resultando num aumento significativo de desempenho e na qualidade do código, pois é possível reutilizar métodos sem duplicar o código específico [85]. Esta técnica foi utilizada nos métodos que processa os pedidos ao API (Código 4, linha 1) e serializa os objetos das diferentes coleções de dados.

```
1.  public static async Task<T> RetrieveDeserializeObject<T>(string baseUrl,
    string getAsyncUrl) where T : class
2.  {
3.      try
4.      {   // ...
5.            deserializeObject = await Task.Run(() =>
                        JsonConvert.DeserializeObject<T>(content));
6.          }
7.          return deserializeObject;
8.       }
9.       catch (Exception){ return null;}
10. }
```

Código 4: Trecho de código da utilização de tipos genéricos e programação assíncrona.

Acrescentado aos pontos acima referidos, a utilização desta técnica também beneficia na reutilização do esforço de implementação. Os tipos e dados podem mudar sem bloqueio de código. É possível desenvolver, testar e alojar o código uma vez, reutilizá-lo com qualquer tipo, incluindo tipos futuros com suporte do compilador e

segurança [85]. A não necessidade do *boxing* (processo de conversão de um tipo de valor para o tipo objeto ou para qualquer tipo de interface implementado por este tipo de valor [86]) e *unboxing* (faz a extração do tipo de valor do objeto [86]) ou o *downcasting* (operação que cria uma referência de subclasse a partir de uma referência de classe base [87]) possibilita melhorar o desempenho do sistema [88].

Para terminar esta secção é importante realçar a utilização de ficheiros de configuração para definir algumas variáveis a serem utilizados ao longo da execução do serviço. Assim sendo, foram definidos no *appSettings* do ficheiro de configuração, pontos finais relativos ao API do motor de *workflows* com o intuito de reutilizar e ter tudo centralizado, facilitando o acesso, configuração e alteração, sempre que necessário.

A *tag appSettings* permite armazenar dados de configuração personalizadas chave/valor, como *connection strings*, *file paths*, URLs ou qualquer informação armazenada no arquivo .ini de uma aplicação [89]. Ganha-se com esta técnica, porque é de fácil manuseio e também porque possui o acesso a leitura e escrita durante o tempo de execução.

Tal como apresentado na secção 4.14, o serviço compositor, também utiliza os mesmos parâmetros de configuração para comportamentos e ligações (Código 2), advindo do comportamento padrão de outros serviços da solução do SIJ.

No trecho de Código 5, é apresentada a definição do serviço compositor no ficheiro de configuração, utilizando os comportamentos definidos e referidos anteriormente, indicada a interface (linha 2) que implementa e o endereço base de acesso ao ponto final do serviço (linha 6).

```
1. <service behaviorConfiguration="DebugModeBehavior"
           name="WorkflowEngineComposer.WorkflowEngineComposerService">
2.     <endpoint
         contract="WorkflowEngineComposer.IWorkflowEngineComposerService"
                binding="wsHttpContextBinding"
                bindingConfiguration="LargeMessageBinding"
                address="" name="WorkflowEngineComposerServiceEndPoint" />
3.     <endpoint address="mex" binding="mexHttpBinding"
                name="metadata" contract="IMetadataExchange" />
4.     <host>
5.       <baseAddresses>
6.         <add
             baseAddress=
   "http://localhost:2005/MJCVServices/WorkflowEngineComposerService/"/>
7.       </baseAddresses>
8.     </host>
9. </service>
```

Código 5: Trecho de código da definição do serviço compositor no ficheiro de configuração.

5.4.2 Alterações nos projetos da solução

No decorrer das adaptações realizadas, foram identificadas alterações a realizar nos membros já existentes, para possibilitar a integração dos novos componentes. Essas alterações, que na maioria dos casos cingiram-se às referências aos serviços, tendo sido adicionadas novas referências correspondentes aos serviços granulares e ao serviço compositor.

Outra alteração que aconteceu, esteve relacionada com a configuração do comportamento do serviço de comunicação. O serviço de comunicação foi um dos componentes onde, teve-se de ser alterados alguns atributos comportamentais na gestão de instâncias do serviço. A gestão de instância dos serviços permite controlar a forma como os objetos dos serviços WCF são instanciados, bem como o tempo que uma instância de serviço deve permanecer no servidor.

Numa comunicação básica (pedido/resposta) a serviços WCF, o cliente faz uma requisição ao objeto do serviço. Este objeto é criado e a instância do serviço WCF atende a solicitação enviando a resposta para o cliente. Desta forma, o serviço de comunicação utilizava apenas uma única instância global para o acesso de todos os clientes [90]. Este designa-se por modo de instância única (Figura 56). Em modo de instância única, o cliente 1 faz uma requisição ao serviço. Uma instância do serviço é criada e a solicitação é atendida. A instância do serviço não é destruída, sendo persistido para o servir a outros pedidos.

Suponha-se, um novo cliente1, cliente 2, que faz uma chamada ao servidor. A mesma instância que foi criada para o cliente 1 é usada para atender a solicitação do cliente 2. Em outras palavras, apenas uma instância global do serviço fornecedor é criada para atender a todos os pedidos do cliente [91].

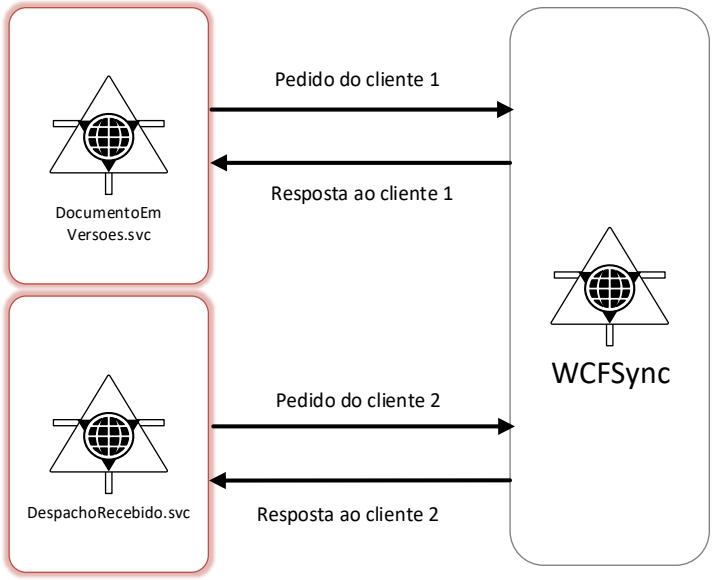


Figura 56: Cenário de instância única, somente uma única instância do serviço será criada [91].

A abordagem anterior de modo de instância única, não adapta as novas características do sistema, uma vez que começaram a acontecer *deadlock* (quando duas ou mais tarefas permanentemente bloqueiam-se, porque o recurso que estão a tentar aceder encontra-se bloqueado por uma outra tarefa [92]), ao aceder ao serviço de comunicação. Para uma melhor compreensão, suponha-se que o cliente 1 faz uma chamada pelo método 1 ao serviço de comunicação, este cria uma única instância para a comunicação entre as partes. Desta forma, impossibilita uma outra chamada do mesmo cliente 1 a partir do método 2, uma vez que mediante a definição do comportamento do serviço de comunicação este só permite a criação de uma única instância por cada chamada a partir de um **único proxy do cliente**. Isto aconteceu sobretudo com os serviços granulares que estão

definidos todos no mesmo projeto e que estão a utilizar **o mesmo proxy para fazer referência ao serviço de comunicação**.

Desta forma, teve de se encontrar uma solução, sendo a que melhor se adaptou a este tipo de cenário, foi a utilização do comportamento modo de instância por sessão (Figura 57) na definição do serviço de comunicação. Este permite manter o estado entre chamadas de métodos ou para uma sessão específica [90]. Neste cenário apenas uma instância de um objeto de serviço de comunicação é criada para uma interação de sessão [91]. Em outras palavras, pode-se dizer que mesmo usando **o mesmo proxy** para referenciar o fornecedor, é possível ter mais de um cliente ou mais de um método por cliente a aceder ao fornecedor sem ocorrências de *deadlock*, porque este cria sessões diferentes para cada chamada de método, independente de estar a utilizar o mesmo proxy ou a partir do mesmo cliente.

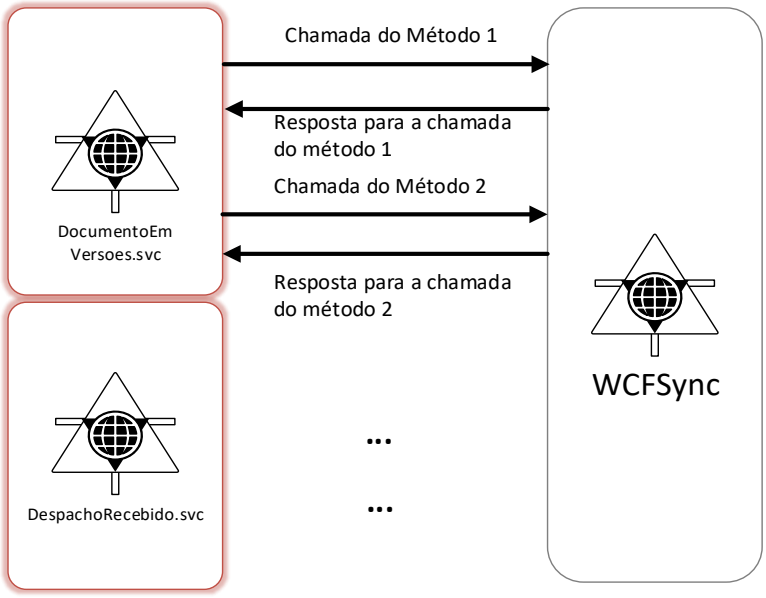


Figura 57: Cenário de instância por sessão, apenas uma instância do serviço será executada para uma sessão de cliente [91].

Assim sendo a interação do serviço com este tipo de comportamento começa eventualmente com o cliente a fazer uma chamada de método. É criada uma instância de serviço, que atende com resposta a chamada de método. É feita uma outra chamada de método na mesma sessão. A mesma instância do serviço serve a chamada de método. Quando o cliente termina a atividade, a instância é destruída sem afetar nenhuma das sessões dos métodos criados estando a concorrência a ser tratado pelos mecanismos de WCF no serviço do fornecedor.

5.5 Execução de testes

A equipa de desenvolvimento encontra-se dividida em três áreas, sendo uma equipa responsável pela camada de apresentação, uma outra para o desenvolvimento da lógica de negócios e uma terceira responsável pelo desenvolvimento de testes do sistema.

Durante o desenvolvimento dos serviços granulares, antes de ser remetido para a equipa de testes, realizaram-se teste manuais, onde estes se destacam por ter como componente chave a interação humana para as tarefas de submissão de dados e também para as tarefas de análise de resultados [93]. Desta forma, foi possível com a realização dos testes manuais, garantir que estes serviços se comportam de acordo com os requisitos definidos nas atividades do sistema de *workflows* do qual possibilitou a sua construção.

Mesmo com os testes manuais realizados e o seu funcionamento de acordo com os requisitos, estes serviços foram remetidos para a equipa de testes a fim de validar, utilizando testes automatizados e teste unitários. Além do serviço granular, outro novo componente do sistema foi alvo de testes unitários automatizados (serviço compositor). Testes a este serviço, basearam-se na interação deste com a API do motor de *workflows*, a fim de verificar se o motor retorna os dados corretos, mediante o modelo visual e os seus metadados.

Apesar da inserção de novos componentes, sabendo que esta nova abordagem difere da anterior tanto na arquitetura como na tecnologia, não foi necessário realizar novos testes de componentes, uma vez que os serviços granulares implementam as mesmas interfaces que o serviço de workflow do processo penal da tecnologia WF. Graças a escalabilidade e modularização no desenvolvimento do sistema e dos testes, não foi necessário implementar novos testes só porque alterou-se componentes. Com isso, obteve-se ganhos que permitiram alocar o tempo e os recursos para a realização de outras tarefas no desenvolvimento do sistema. Assim sendo, os testes de componentes permitiram garantir o igual comportamento com os *workflows* WF anterior.

Por fim, tendo os componentes testados unitariamente e a sua integração módulo a módulo, passou-se a testes automatizados já desenvolvidos, que permite avaliar o correto funcionamento na íntegra do sistema, simulando as ações que os utilizadores podem realizar nas interfaces de uma aplicação. Estes são conhecidos por testes de sistemas, sendo uma delas o "UI Tests" [93].

5.6 Resumo

Neste capítulo foi apresentada a nova abordagem, assente numa nova notação estabelecida e na integração de novas ferramentas que auxiliam no desenvolvimento e execução do sistema de *workflows*, desde o conversor de definição de processos a API do motor de *workflows*.

Seguidamente abordou-se a constituição da nova arquitetura do sistema com a inserção do componente principal e foco deste trabalho, os serviços granulares. Os serviços granulares em conjunto com o novo motor de *workflows* vieram suprir a tecnologia WF permitindo reduzir alguns dos problemas que motivaram essa reestruturação.

Para finalizar o capítulo, apresentaram-se as principais alterações efetuados nos projetos e componentes já existentes da solução. Destaca-se a inserção do novo serviço, designado de compositor que tem como principal objetivo orquestrar as requisições efetuadas pela API do motor de *workflows* e também se apresentou a alteração da forma como são geridas as instâncias do serviço de comunicação. O WCF disponibiliza 3 tipos de gestão de instâncias de serviços, por sessão, simples e por chamada. Na abordagem anterior, utiliza-se a gestão de instância única, ao qual com as alterações a nova arquitetura, deparou-se problemático, em certos momentos

começaram a acontecer *deadlocks* no acesso ao serviço. Assim, este foi alterado, tornando-o por sessão o que se demostrou eficiente e adaptável a nova arquitetura. Igualmente abordou-se a execução dos testes onde se destacam testes unitários, automatizados e testes de integração que permitem validar a qualidade do *software*, garantindo o igual comportamento aos requisitos definidos na abordagem anterior.

6. Conclusão

A dissertação centrou-se na reestruturação do sistema de *workflows* do Sistema de Informação da Justiça de Cabo Verde (SIJ). Esta restruturação permitiu a integração de novas ferramentas para a modelação e definição dos processos de negócios, a integração do novo motor de *workflows*, a supressão de componentes e tecnologias e a reutilização de códigos anteriores, para a criação de serviços granulares e independentes a partir da retirada da lógica de dentro de *workflows*, disponibilizando interfaces de serviços possíveis de serem acedidos independentemente da plataforma, localização ou ambiente que os rodeia.

Com a mudança de paradigma, os sistemas têm sido igualmente desenhados e implementados com auxílio a lógica do processo realizada pela aplicação, facilitando a tarefa de modificação da lógica de negócio, dado que se consegue percecionar visualmente o impacto das alterações e não há necessidade de alteração de código funcional. Advindo disso, cada vez mais os sistemas têm hospedados componentes de *workflow* que facilitam a modelação de processos de negócios conhecido por sistemas de *workflows* integrados. Os sistemas de *workflows* integrados permitem modelar e automatizar etapas em um processo específico, coordenando as interações entre pessoas e sistemas de *software*. Desta forma, é preciso dotar esses sistemas de ferramentas que auxiliam na modelação à execução de *workflows*. A literatura enfatiza que a modelação deve responder a algumas questões na sua implementação, tais como o resultado esperado, as atividades que serão realizadas, as ordens e quem realiza as atividades e igualmente que devem ser identificadas as fraquezas e como estas serão estendidos no futuro. Em adição, é necessário a esses modelos clareza, precisão e devem ser compreensíveis.

Esses modelos e a definição de processos são conseguidos em tempo de construção do *workflow*. A definição de processos permite traduzir processo de negócios do mundo real para uma definição formal compreensível para computadores. Por outro lado, em tempo de execução, a definição do processo é interpretada pelo motor de *workflows*, com competências para criar e controlar instâncias do processo. Esse controlo permite orquestrar as atividades, incorporando as vantagens do paradigma orientado a serviços, onde o componente central (serviços) permite montar sistemas a partir de módulos simples e reutilizáveis, proporcionando a interligação dos processos de negócios com os processos técnicos. Este componente central deve ser explícito, autônomo, partilhando contratos e permitindo a compatibilidade baseada na política.

Como referido ao longo desse estudo, este processo de reengenharia advém de problemas decorrentes no sistema de *workflows* do SIJ. Assim sendo, pretende-se com essa reestruturação evitar e suprir problemas sentidos, assente na reinvenção, onde se destacam a supressão e inserção de novos componentes.

A supressão incluiu a tecnologia WF e serviços implementados nesta tecnologia com auxílio do WCF. Esta supressão como retratado nas secções 1.2 e 3.3, deveu-se à evolução da tecnologia e à incapacidade dessa evolução responder às necessidades do SIJ. Avindo disso, tornou-se urgente arranjar alternativas e novas abordagens para a concessão e manutenção do sistema de *workflows* do SIJ.

Nesse contexto, estabeleceu-se uma notação assente em UML para a modelação dos processos e *workflows* (secção 5.1). Com esta nova notação foi possível modelar os *workflows* que se encontram desenhados na

ferramenta embutida do WF (secção 5.1.2). A modelação numa primeira fase abarca apenas os processos penais e os *workflows* auxiliares, onde com estes modelados, foi possível convertê-los para uma definição compreensível ao motor de *workflows* na secção 5.2. Tanto o motor, como o conversor foram desenvolvidos de zero pela equipa de desenvolvimento do SIJ, tendo sempre em mente as necessidades e problemas que afetam o sistema. Foram inúmeras as reuniões entre a equipa para a definição dos requisitos usados na concessão do motor, conversor e no estabelecer da notação. Com as ferramentas desenvolvidas e apto para auxiliar no desenvolvimento, execução e manutenção do sistema de *workflow*, passou-se a reutilização do código contido nos *workflows* anteriores. A reutilização permitiu criar serviços granulares, podendo ser acedidos independente da localização (Capítulo 4). Estes serviços contêm a lógica central do SIJ para processos penais. Para a sua integração foi necessário alterar diversos componentes existentes. Muitas dessas alterações acabaram por ser cirúrgicas, não obstante, outras mais complexas tiveram de ser efetuados para adaptação à nova arquitetura. Das alterações mais complexas, destaca-se a inserção do serviço compositor, que orquestra as interações da solução do SIJ com a API do motor de *workflows* e a alteração do modo de gestão da instância do serviço de comunicação.

No final dessa restruturação incluindo a supressão, inserção e reutilização de componentes e tecnologias tem-se o "novo" sistema de *workflows* do SIJ capaz de ser escalável, dinâmico e que otimiza a lógica de negócio por detrás da tramitação e desmaterialização dos processos nos tribunais de Cabo Verde.

6.1 Trabalho futuro

Esse estudo baseou-se nos *workflows* do processo penal ordinário, sumario, abreviado e transação. O sistema judicial é composto por vários outros tipos de processos tais como processos cíveis, processo penal *habeas corpus*, recursos e outros. A modelação dos *workflows* do processo cível encontra-se separada por processo civil ordinário, providências cautelares não especificadas, providências cautelares de alimentos provisórios, providências cautelares de arresto, providências cautelares de embargo, providências cautelares de arrolamento, ações executivas de penhora, embargos à execução, embargos à providência entre outros. O recurso pode ser ordinário ou extraordinário sendo ordinário de apelação, revista e extraordinário de revisão.

Desta forma, tal como o processo penal, pretende-se alargar essa nova abordagem de suprir a tecnologia anterior e retirar a lógica de dentro de *workflows* para serviços granulares também a esses tipos de processos cíveis e recursos. Para cada um desses tipos de processos será necessária a referente modelação e automatização do *workflow* assente na nova notação e na nova abordagem de ter serviços granulares. Já o processo penal de *habeas corpus* será *workflow* único, sendo menos complexos do que os outros. Tal como o processo penal anterior, todos os processos do sistema se encontram modelados e usando a tecnologia WF, assim sendo, a metodologia a ser usada para esses outros processos serão semelhantes ao utilizado neste estudo.

Advindo da passagem desses restantes tipos de *workflow* para esta abordagem, torna-se possível simplificar ainda mais a arquitetura (Figura 58). Esta simplificação passa pela supressão do serviço de comunicação, permitindo a interação direta e total entre o controlador e os serviços granulares. Devido à limitação temporal

para a realização deste trabalho e o importante papel desempenhado na interligação dos outros serviços de *workflow* que ainda permanecem na tecnologia WF, a retirada do serviço de comunicação tornou-se inviável.

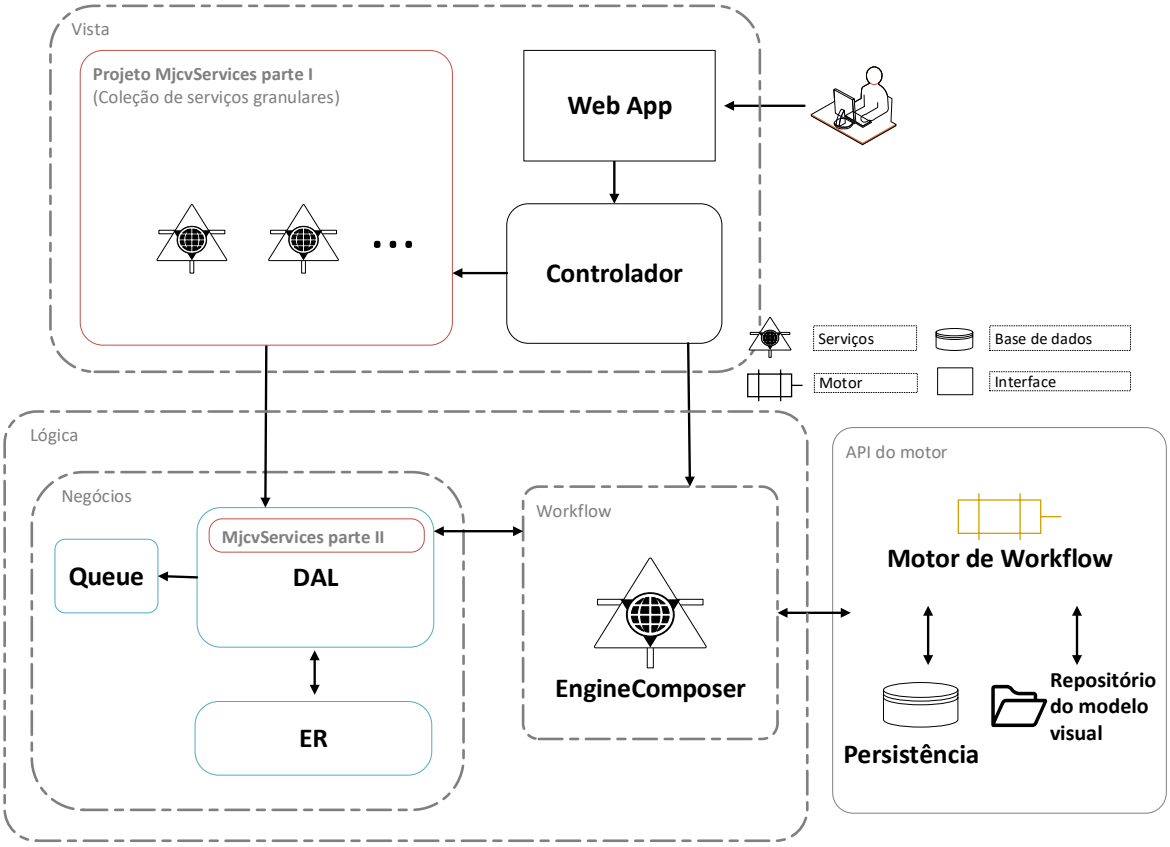


Figura 58: A arquitetura depois da supressão do serviço de comunicação.

Outro ponto que se torna pertinente asseverar futuramente para a consolidação e uniformização do sistema, tem a ver com garantir a implementação de mecanismos que permitam a integridade e direitos de acesso as interfaces dos serviços granulares a quem tem perfil para tal, em tempo de execução do *workflow*. O acesso a recursos nesses serviços granulares fracamente acoplados, deve ser autorizado apenas a perfis de utilizadores que tenham permissão mediante a fase que o processo se encontre, impossibilitando assim a qualquer um que tenha acesso a URL poder aceder ou efetuar ações mesmo sem o perfil e privilégio adequado. Futuramente pretende-se fazer o uso do padrão OAuth ou da autorização baseada em *Tokens*. Esses dois mecanismos podem adaptar-se ao sistema, no sentido que o OAuth pode ser utilizado para garantir o acesso aos recursos dos serviços granulares que não possuem a proteção lógica, sendo necessários mecanismos mais robustos de autenticação e a autenticação baseada em *tokens* será implementado na API do motor do *workflow*, blindando a API com mais um mecanismo de segurança, sendo esta privada, tendo já mecanismos de segurança implementados na infraestrutura que o protege.

Bibliografia

[1] B. Bruce, *Pro WF: Windows Workflow in .NET 3.5*. 2010.

[2] B. White, *Pro WF 4.5*, vol. 53. 2013.

[3] D. Esposito, "Getting Started with Microsoft Windows Workflow Foundation: A Developer Walkthrough," *Solid Quality Learning*, 2006. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa480214.aspx.

[4] D. Chappell, "Introducing Microsoft Windows Workflow Foundation: An Early Look," *Chappell & Associates*, 2005. [Online]. Available: https://msdn.microsoft.com/en-us/library/aa480215.aspx.

[5] M. Milner, "A Developer's Introduction to Windows Workflow Foundation (WF) in .NET 4," *Pluralsight*, 2009. [Online]. Available: https://msdn.microsoft.com/en-us/library/ee342461.aspx.

[6] M. Liu, *WCF 4.5 Multi-Layer Services Development with Entity Framework Third Edition*. 2012.

[7] M. Juric and K. Pant, *Business Process Driven SOA Using BPMN and BPEL: From Business Process Modeling to Orchestration and Service Oriented Architecture*. 2008.

[8] "Documentação Interna do SIJ - Sistema de Informação da Justiça de Cabo Verde."

[9] Microsoft IC, "Overview of the .NET Framework." [Online]. Available: https://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.110).aspx.

[10] Microsoft IC, "Endpoints: Addresses, Bindings, and Contracts," 2017. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/endpoints-addresses-bindings-and-contracts.

[11] Microsoft IC, "What Is Windows Communication Foundation." [Online]. Available: https://msdn.microsoft.com/en-us/library/ms731082(v=vs.110).aspx.

[12] Microsoft IC, "Introduction to Building Windows Communication Foundation Services." [Online]. Available: https://msdn.microsoft.com/en-us/library/aa480190.aspx.

[13] Microsoft IC, "Introducing Windows Communication Foundation in .NET Framework 4." [Online]. Available: https://msdn.microsoft.com/en-us/library/ee958158.aspx.

[14] A. Sharp and P. Mcdermoot, *Workflow Modeling: Tools for Process Improvement and Applications Development 2nd Edition*, vol. 104, no. 25. 2009.

[15] P. Harmon, *Business Process Change: A Guide for Business Managers and BPM and Six Sigma Professionals 2nd Edition*. 2007.

[16] M. Laguna and J. Marklund, *Business Process Modeling, Simulation And Design 2nd Edition*. 2013.

[17] M. Hewing, *Business Process Blueprinting: A Method for Customer-Oriented Business Process*

Modeling. 2014.

[18] M. Weske, *Business Process Management: Concepts, Languages, Architectures 2nd Edition*. 2012.

[19] M. Weske, *Business Process Management: Concepts, Languages, Architectures*. 2007.

[20] P. Desfray and G. Raymon, *Modeling Enterprise Architecture with TOGAF: A Practical Guide Using UML and BPMN*. 2014.

[21] V. Stiehl, R. Raw, and P. Smith, *Process-Driven Applications with BPMN*. 2014.

[22] A. Iyengar, V. Jessani, and M. Chilanti, *WebSphere Business Integration Primer: Process Server, BPEL, SCA and SOA*. 2008.

[23] D. Minoli, *Enterprise Architecture A to Z: Frameworks, Business Process Modeling, SOA, and Infrastructure Technology*. 2008.

[24] Y. Chen-Burger and D. Robertson, *Automating Business Modelling: A Guide to Using Logic to Represent Informal Methods and Support Reasoning*. 2005.

[25] S. White and D. Miers, *BPMN Modeling and Reference Guide: Understanding and Using BPMN*. 2008.

[26] Creately, "Business Process Modeling Techniques with Examples," 2014. [Online]. Available: http://creately.com/blog/diagrams/business-process-modeling-techniques/#uml.

[27] R. Shapiro *et al., BPMN 2.0 Handbook Second Edition: Methods, Concepts, Case Studies and Standards in Business Process Modeling Notation (BPMN)*. 2012.

[28] Object Management Group (OMG), "Business Process Model and Notation (BPMN) Version 2.0," *Business*, vol. 50, no. January, p. 170, 2011.

[29] O. M. G. D. Number and P. D. F. A. File, "BPMN 2.0 by Example," vol. 0, no. June, 2010.

[30] E. Herrera, *The BPMN Graphic Handbook*. 2015.

[31] Kurt Jensen and L. M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. 2009.

[32] F. Schönthaler, G. Vossen, A. Oberweis, and T. Karle, *Business Processes for Business Communities: Modeling Languages, Methods, Tools*. 2012.

[33] M. Von Rosing, H. Von Scheel, and A. W. Scheer, *The Complete Business Process Handbook: Body of Knowledge from Process Modeling to BPM*, vol. 1. 2014.

[34] "Object oriented methodologies," 2012. [Online]. Available: https://www.slideshare.net/naina-rani/object-oriented-methodologies.

[35] W. Zhao, A. Wang, and X. Fang, "Role-Activity Diagrams Modeling Based on Workflow Mining," 2014. [Online]. Available: https://www.slideshare.net/supercornetto/role-activity-diagrams-modeling-based-on-workflow-mining?qid=a9ed57e9-509c-4ed3-a38a-

55da170795d1&v=&b=&from_search=1.

[36] P. Harmon, *Business Process Change: A Business Process Management Guide for Managers and Process Professionals 3rd Edition*, vol. 53. 2014.

[37] M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers, *Fundamentals of Business Process Management*. 2012.

[38] W. M. P. Van Der Aalst and K. M. van Hee, *Workflow Management Models, Methods, and Systems*. 2004.

[39] D. Hollingsworth, "Workflow Management Coalition: The Workflow Reference Model," *WfMC Specif.*, no. 1, pp. 1–55, 1994.

[40] D. M. R. Ferreira, *Workflow Management Systems Supporting the Engineering of Business Networks*. 2003.

[41] M. Havey, *Essential Business Process Modeling*, no. August. 2005.

[42] C. G. Schuetz, *Multilevel Business Processes: Modeling and Data Analysis*. 2015.

[43] James F. Chang, *Business Process Management Systems: Strategy and Implementation*. 2004.

[44] C. M. Mackenzie, K. Laskey, F. Mccabe, R. Metz, and A. Hamilton, "Reference Model for Service Oriented Architecture 1.0 Committee Specification 1, 2 August 2006," no. August, pp. 1–31, 2006.

[45] M. Gousset, B. Keller, A. Krishnamoorthy, and M. Woodward, "Professional WCF 4 Windows Communication Foundation with NET 4," 2010.

[46] M. Liu, *WCF 4.0 Multi-tier Services Development with LINQ to Entities*, no. 4. 2010.

[47] N. Pathak, "Pro WCF 4," pp. 3–24, 2011.

[48] J. Lowy, *Programming WCF Services*. 2010.

[49] M. L. Bustamante, *Learning WCF, Master Windows Communication Foundation and SOA Fundamentals*. 2007.

[50] Scott Klein, *Professional WCF Programming*. 2007.

[51] "10 Principles of SOA," 2007. [Online]. Available: https://www.infoq.com/articles/tilkov-10-soa-principles.

[52] C. Interfaces, "Common Object Request Broker Architecture (CORBA) Specification, Version 3.3," no. November, 2012.

[53] B. S. of I. University of California, "Distributed Computing Applications and Infrastructure." [Online]. Available: http://courses.ischool.berkeley.edu/i206/f97/GroupB/mom/what_is_mom.html.

[54] "The OAuth 2.0 Authorization Framework rfc6749," 2012. [Online]. Available: https://tools.ietf.org/html/rfc6749.

[55] "Cookies vs Tokens: The Definitive Guide," 2016. [Online]. Available: https://auth0.com/blog/cookies-vs-tokens-definitive-guide/.

[56] J. Rosa, "Aplicaçao de Sistemas de Workflow em Cenários de E-Government," p. 82, 2012.

[57] "Documentação Interna do SIJ - SIPP." .

[58] I. S. N. O. B. O. Da, R. D. E. Cabo, and V. D. E. Novembro, "Código de Processo Penal da República de Cabo Verde," 2015.

[59] "Entity Framework Overview," 2017. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/ef/overview.

[60] "Entity Framework." [Online]. Available: https://docs.microsoft.com/en-us/ef/.

[61] Microsoft IC, "Understanding WSDL," 2003. [Online]. Available: https://msdn.microsoft.com/en-us/library/ms996486.aspx.

[62] W3C, "Web Services Description Language (WSDL) 1.1," 2001. [Online]. Available: https://www.w3.org/TR/wsdl.

[63] M. Fowler, K. Beck, J. Brant, W. Opdyke, and D. Roberts, "Refactoring: Improving the Design of Existing Code," *Xtemp01*, vol. 201485672, pp. 1–337, 1999.

[64] M. Fowler, "Yagni," 2015. [Online]. Available: https://martinfowler.com/bliki/Yagni.html.

[65] B. R. Myers, *Foundations of WF An Introduction to Windows Workflow Foundation*. 2007.

[66] B. Bukovics, *Pro WF: Windows Workflow in .NET 3.0*. 2007.

[67] "What is a DLL?" [Online]. Available: https://support.microsoft.com/en-gb/help/815065/what-is-a-dll.

[68] P. K. Katiyar, "Understanding Contracts in WCF," 2013. [Online]. Available: https://www.codeproject.com/Articles/664238/Understanding-Contracts-in-WCF.

[69] R. Crane, S. Resnick, and C. Bowen, *Essential Windows Communication Foundation (WCF)*. 2008.

[70] M. Liu, *WCF Multi-layer Services Development with Entity Framework - Fourth Edition*. 2014.

[71] ".VSDX File Extension." [Online]. Available: https://fileinfo.com/extension/vsdx.

[72] "Django documentation." [Online]. Available: https://docs.djangoproject.com/en/1.11/.

[73] The PostgreSQL Global Development Group, "PostgreSQL 9.6.3 Documentation."

[74] "HTTP Methods: GET vs. POST." [Online]. Available: https://www.w3schools.com/tags/ref_httpmethods.asp.

[75] "Introduction to Windows Service Applications," 2007. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/windows-services/introduction-to-windows-service-applications.

[76] "JSON-LD 1.0." [Online]. Available: https://www.w3.org/TR/json-ld/#introduction.

[77] "Side Effects," 2016. [Online]. Available: https://docs.microsoft.com/en-us/cpp/c-language/side-effects.

[78] "SQL Server UNIQUEIDENTIFIER data type." [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSBJG3_2.5.0/com.ibm.gen_busug.doc/c_fgl_odiagmsv_007.htm.

[79] "RFC 2617 HTTP Authentication," 1999. [Online]. Available: https://tools.ietf.org/html/rfc2617.

[80] "JSON Web Token (JWT)," 2015. [Online]. Available: https://tools.ietf.org/html/rfc7519#section-1.

[81] "Asynchronous Programming with Async and Await (C# and Visual Basic)." [Online]. Available: https://msdn.microsoft.com/library/hh191443(vs.110).aspx.

[82] "Asynchronous programming," 2016. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/async.

[83] "Async in depth," 2016. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/standard/async-in-depth.

[84] "Introduction to Generics (C# Programming Guide)." [Online]. Available: https://msdn.microsoft.com/en-us/library/0x6a29h6(v=vs.120).aspx.

[85] "An Introduction to C# Generics." [Online]. Available: https://msdn.microsoft.com/en-us/library/ms379564(v=vs.80).aspx#csharp_generics_topic9.

[86] Microsoft IC, "Boxing and Unboxing (C# Programming Guide)," 2017. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/boxing-and-unboxing.

[87] "Downcasting in C# - C# Corner," 2009. [Online]. Available: http://www.c-sharpcorner.com/uploadfile/scottlysle/downcasting-in-C-Sharp/.

[88] "Benefits of Generics (C# Programming Guide)." [Online]. Available: https://msdn.microsoft.com/en-us/library/b5bx6xee(v=vs.120).aspx.

[89] "<appSettings> Element." [Online]. Available: https://msdn.microsoft.com/en-us/library/aa903313(v=vs.71).aspx.

[90] "Concurrency Management." [Online]. Available: https://msdn.microsoft.com/en-us/library/orm-9780596521301-02-08.aspx.

[91] S. Koirala, "Three ways to do WCF instance management," 2010. [Online]. Available: https://www.codeproject.com/Articles/86007/ways-to-do-WCF-instance-management-Per-call-Per.

[92] "Deadlocking." [Online]. Available: https://technet.microsoft.com/en-us/library/ms177433(v=sql.105).aspx.

[93] J. A. Brazeta, "Testes de Software no Sistema de Informação da Justiça Cabo-Verdiana," 2014.

Índice dos principais termos

Anexo A – Modelo conceptual do Wf. Proc. Penal